UNIVERSIDAD AUTÓNOMA METROPOLITANA

Unidad Cuajimalpa

Ingeniería en Computación

Arquitectura de computadoras

Dr. Luis Ángel Alarcón Ramos

Procesador Tipo Harvard

Saucedo Mendoza Santiago

Cervantes Vázquez Jonathan Omar

Zaragoza Valle Diego

Hernández Landeros Ivanna Valeria

Índice:

1. Introducción y descripción general

1.1 Introducción

El presente trabajo se centra en el diseño y desarrollo de un procesador de 8 bits, que cuenta con un conjunto de instrucciones que incluyen: operaciones aritméticas y lógicas, transferencia de datos, saltos condicionales e incondicionales.

Las características principales del procesador son:

- Arquitectura Harvard de 8 bits para datos.
- Un conjunto de 32 instrucciones.
- 2 modos de direccionamiento (directo y por registro).
- Memoria de datos y memoria de programa separada.
- Un conjunto de registros de almacenamiento para gestionar la ejecución de las instrucciones.
- Palabra de instrucción.

1.2 Planteamiento del proyecto

En este trabajo se desea implementar un procesador de 8 bits utilizando el lenguaje de descripción de hardware VHDL.

El diseño ejecutará una instrucción en cada ciclo de reloj. Tendrá una organización de memoria parecida a la arquitectura Harvard, para poder manejar por separado la memoria de datos y la memoria del programa.

El programa en lenguaje ensamblador y su equivalente en lenguaje máquina estará colocado en la memoria del código para su ejecución.

Desarrollar el microprocesador representa un ejercicio fundamental en el aprendizaje de arquitectura de computadoras, ya que se integran conceptos teóricos en una implementación práctica.

Enfoque en tres pilares fundamentales:

- Diseño conceptual: Arquitectura Harvard, conjunto de 32 instrucciones y un sistema de direccionamiento dual.
- Implementación técnica: Integración de unidades funcionales, mecanismos de control y sincronización, y finalmente el modelado preciso en VHDL.

- Validación de resultados: Simulación de operaciones, pruebas con programas prácticos y por último el análisis de comportamientos.

1.3 Objetivos del proyecto

Diseñar e implementar un procesador tipo Harvard de 8 bits y con un conjunto de 32 instrucciones, utilizando 2 modos de direccionamiento (directo y por registro) utilizando el lenguaje de descripción de hardware VHDL.

- Diseño del conjunto de instrucciones

1.4. Limitaciones

Este proyecto está limitado en los siguientes aspectos:

- La longitud de las posibles combinaciones para las operaciones de datos es de 8 bits.
- El diseño del procesador se limita a la ejecución de instrucciones de un solo ciclo.

2. Marco teórico

Los elementos básicos de un procesador son una unidad aritmética y lógica (ALU), una memoria de instrucciones, una memoria de datos y una unidad de control que es la encargada de supervisar la manera en la que circulan las señales de entrada y salida entre otros componentes.

Desde el punto de vista del lenguaje ensamblador, la arquitectura de una computadora, en este caso de un procesador, se describe a partir de su conjunto de instrucciones, que incluyen el código de operación, registros, modos de direccionamiento, etc.

2.1. Arquitectura Harvard

La arquitectura tipo Harvard, utiliza memorias separadas para instrucciones y datos. La memoria de programa almacena instrucciones, por lo que la memoria deberá ser solamente de lectura, con sus propios buses de direcciones, instrucciones y control.

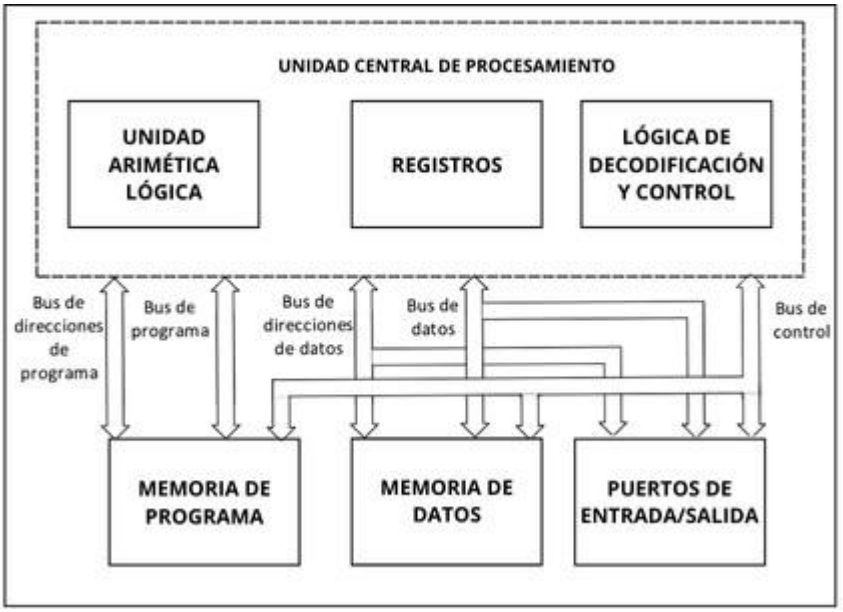


Figura 2.1. Computadora tipo Harvard.

2.2. Lenguaje VHDL

En VHDL se hace la descripción de hardware, esto permite modelar sistemas digitales en un nivel estructural y comportamental, también simula el comportamiento de los diseños antes de una implementación física y finalmente sintetiza código de FPGAs.

El método de programación implementado en este proyecto es estructural, esto significa que: se describe el circuito con instancias de componentes. Estas instancias forman un diseño de jerarquía superior, al interconectar los puertos de estas instancias con las señales internas del circuito, o con puertos del circuito de jerarquía superior.

Se propone dividir el sistema en bloques, donde se caracterizan los componentes básicos del circuito y con estos se forman bloques de mayor tamaño que representen un circuito más complejo, que luego se enlazan para formar componentes más grandes, que a su vez se enlazan para formar el sistema completo.

Las operaciones ocurren de manera concurrente, esto refleja el comportamiento paralelo utilizado en hardware real, tiene una descripción jerárquica ya que permite dividir el diseño del microprocesador en módulos independientes los cuales se interconectan mediante señales.

2.3 Conjunto de instrucciones

Instrucciones				
Instrucciones aritméticas:	**Instrucciones lógicas:**	**Instrucciones de transferencia**	**Instrucciones de saltos**	**Instrucciones de comparación**
ADD	AND	MOV	JMP	CMP
ADC	OR	LD I	JC/JNC	
SUB	XOR	ST	JS/JNS	
INC	NAND	LD	JV/JNV	
DEC	NOR		JZ/JNZ	
SHR	XNOR			
SHL	NOT			
	XOR			
	ROL			

Tabla .2.3.1. Conjunto de instrucciones.

2.4 Modos de direccionamiento

- **Direccionamiento por registro:** Los operandos están contenidos en registros, la operación solicitada indica cuáles registros se deben utilizar, después se accede directamente al contenido de dichos registros.

  Ejemplo: MOV A, B (copia el contenido del registro B al registro A).

- **Direccionamiento inmediato:** El operando es un valor constante el cual es incluido directamente en la instrucción. En este modo no se accede a registros para obtener dicho operando, porque se utiliza directamente en la instrucción.

  Ejemplo: MOV A, 8 (inserta el valor 8 directamente en el registro A).

3. Diseño del microprocesador

3.1. Arquitectura del procesador

La arquitectura general del procesador se estructura en dos módulos principales: el Datapath y la Unidad de Control (UC), los cuales trabajan de forma coordinada.

Datapath

El Datapath se encarga de realizar las operaciones de procesamiento de datos que la Unidad de Control le indica a través de una **palabra de control (CW)**. El Datapath está compuesto por:

- **Unidad de Registros (UR):**
  Contiene **8 registros** (R0 a R7).
- **Unidad Funcional (UF):**
  La UF se subdivide en tres bloques principales:
  - **Unidad Aritmética (UA):**
    Implementada mediante un sumador de 8 bits, responsable de las operaciones de suma (ADD), suma con acarreo (ADC), resta (SUB), incremento (INC) y decremento (DEC). La UA también contempla el manejo del bit de acarreo (C).
  - **Unidad Lógica (UL):**
    Implementada utilizando un multiplexor que canaliza los resultados de operaciones lógicas como AND, OR, XOR, NAND, NOR, XNOR, y NOT entre operandos.
  - **Shifter:**
    Implementado mediante multiplexores que permiten realizar desplazamientos lógicos a la izquierda (SHL) o derecha (SHR), así como rotaciones (ROL y ROR) de un bit.
- **Registro de Estado:**
  Registra las banderas de estado **C (acarreo)**, **S (signo)**, **V (overflow)** y **Z (cero)**, las cuales se actualizan en operaciones específicas para su uso en instrucciones condicionales.

- **Registros Especiales:**
    - **PC (Contador de Programa):**
      Almacena la dirección de la instrucción actual en ejecución.

    - **IR (Registro de Instrucción):**
      Guarda la instrucción leída desde la memoria del programa..

    - **CW (Palabra de Control):**
      Contiene las señales de control necesarias para coordinar las operaciones internas del procesador en cada ciclo.

Unidad de Control (UC)

La Unidad de Control interpreta el contenido del Registro de Instrucción (IR) y genera la Palabra de Control (CW) adecuada para dirigir al Datapath. Se compone de:

- **Decodificador:**
  Transforma el código de operación contenido en el IR en una serie de señales de control específicas.
- **Secuenciador:**
  Basado en una **carta ASM** (Algorithmic State Machine), gestiona la transición de estados del procesador y controla la actualización del PC en operaciones secuenciales y saltos.
  Unidad de Salto:
  Evalúa las condiciones necesarias (basadas en las banderas de estado) para realizar saltos incondicionales (JMP) o condicionales (JC, JNC, JS, JNS, JV, JNV, JZ, JNZ) durante la ejecución del programa.

Memorias

Siguiendo la arquitectura Harvard, el procesador utiliza dos memorias separadas:

- **Memoria de Código:**
  Contiene las instrucciones del programa a ejecutar.

- **Memoria de Datos:**

  Almacena los datos necesarios para la ejecución del programa, así como los resultados generados por las instrucciones.

Modos de Direccionamiento

- Modo directo
- Modo por registro

3.2. Unidad aritmética-lógica (ALU)

Especificaciones técnicas:

- **Entradas principales:**
  - **A[7:0], B[7:0]: Operandos**
  - **S[3:0]: Selector de operación (bit 3: UA/UL, bits 2-0: código operación)**
  - **carry_in: Acarreo de entrada**
- **Salidas:**
  - **R[7:0]: Resultado**
  - **Banderas: Carreo, Vover, Ze, Sig**

Tabla de operaciones

S[3]	S[2:0]	Operación	Tipo
0	000	ADD	UA
0	001	SUB	UA
0	110	ADC	UA
1	000	AND	UL
1	001	OR	UL
1	010	NOT A	UL

1	011	XOR	UL
1	100	NOR	UL
1	101	NAND	UL
1	110	XNOR	UL
1	111	NOT B	UL

Tabla 3.1.1. Tabla de operaciones de la ALU.

Módulos:

1. **ALU_UA_F (Unidad Aritmética):**
    - **Implementa sumas/restas con manejo de acarreo**
    - **Genera 4 banderas:**
        - **carry: Acarreo**
        - **vout: Overflow**
        - **zero: Zero (resultado = 0)**
        - **signo: Signo (bit más significativo)**
2. **ALU_UL (Unidad Lógica):**
    - **Opera en modo bit-a-bit**
    - **Implementa 8 operaciones mediante multiplexor 8:1**
    - **Genera banderas zero y signo**
3. **Mux2a18Bits:**
    - **Selecciona entre resultado aritmético (UA) o lógico (UL)**
    - **Controlado por S[3]**

Tabla de activación por operación de las banderas:

Operación	**Carry**	**Overflow**	**Zero**	**Signo**
ADD	**Sí**	**Sí**	**Sí**	**Sí**

ADC	**Sí**	**Sí**	**Sí**	**Sí**
SUB	**Sí**	**Sí**	**Sí**	**Sí**
AND	**No**	**No**	**Sí**	**Sí**
OR	**No**	**No**	**Sí**	**Sí**
XOR	**No**	**No**	**Sí**	**Sí**
NOT	**No**	**No**	**Sí**	**Sí**

Tabla .3.3.2. Conjunto de instrucciones.

Las banderas aritméticas y lógicas se generan en paralelo y es el bit S(3) del selector decide cuáles son válidas.

3.3. Unidad de Salto

La unidad de salto determina si se debe realizar un salto condicional basado en las banderas de estado (Carry, Signo, V, Zero) y el código de operación (S[2:0]). Genera una señal SV (Salto Válido) que indica si se cumple la condición de salto.

Lógica de Operación:

- **Entradas**:
    - Banderas:
        - Carry: Acarreo (C)
        - Signo: Signo (S, bit más significativo)
        - V: Overflow (V)
        - Zero: Resultado cero (Z)
    - Selector S[2:0]: Código de condición de salto.
- **Salida**:
    - SV:

- 1 → Salto válido (se cumple la condición).
- 0 → No saltar.

S[2:0]	Condición	Lógica	Instrucción
000	Carry = 1	SV = Carry	JC (Jump si Carry = 1)
001	Carry = 0	SV = NCarry	JNC
010	Signo = 1	SV = Signo	JS (Jump si Signo = 1)
011	Signo = 0	SV = NSigno	JNS
100	Overflow = 1	SV = V	JV (Jump si Overflow = 1)
101	Overflow = 0	SV = NV	JNV
110	Zero = 1	SV = Zero	JZ (Jump si Zero = 1)
111	Zero = 0	SV = NZero	JNZ

Tabla .3.3.3. Tabla unidad de salto.

Implementación Estructural (Ecuaciones):

- **Genera las señales negadas de las banderas:**

```
NOT_A_0: NOT_A port map(a => Carry, r => NCarry);
NOT_A_1: NOT_A port map(a => Signo, r => NSigno);
NOT_A_2: NOT_A port map(a => V,r => NV);
NOT_A_3: NOT_A port map(a => Zero, r => NZero);
```

- **Selecciona la bandera (o su negación) según S[2:0]:**

```
Mux8a1_SV: Mux8a1 port map(
    a => Carry, b => NCarry, c => Signo, d => NSigno,
    e => V, f => NV, g => Zero, h => NZero,
    S => S, r => SV);
```

3.4. Comparador

El propósito del comparador de 8 bits es verificar si dos operandos (A y B) son iguales, generando una señal CMP en nivel alto (1) cuando son idénticos, y en bajo (0) cuando son distintos.

Lógica de Operación

- **Entradas:**
  - **A[7:0], B[7:0]: Operandos a comparar.**
- **Salida:**
  - **CMP:**
    - **1 si A = B (los bits son iguales).**
    - **0 si A ≠ B (al menos un bit es diferente).**

Ecuación:

CMP <= NOT ((A(0) XOR B(0)) OR
(A(1) XOR B(1)) OR
(A(2) XOR B(2)) OR
(A(3) XOR B(3)) OR
(A(4) XOR B(4)) OR
(A(5) XOR B(5)) OR
(A(6) XOR B(6)) OR
(A(7) XOR B(7)))

Implementación Estructural:

1. **XOR Bit a Bit**:
   - Compara cada par de bits (A(i) vs B(i)).
   - Resultado intermedio: 1 si los bits difieren, 0 si son iguales.
2. **OR de 8 Bits**:
   - Agrega todos los resultados XOR. Si *al menos un bit es diferente*, la salida del OR será 1.
3. **NOT Final**:
   - Invierte la salida del OR:
     - OR=1 (hay diferencias) → CMP=0.

- OR=0 (todos iguales) → CMP=1.

3.5. Registro IR (Instruction Register)

Componente clave en la arquitectura del microprocesador. La función principal es almacenar temporalmente la instrucción que está siendo ejecutada, gracias a esto permite su decodificación y posterior procesamiento.

Es un registro de instrucción con almacenamiento sincronizado por reloj y utiliza un control de escritura(utiliza flip-flops tipo D con enable).

- Código de operación (S_out)
- Direcciones de registros (DA_out, DB_out, DC_out)
- Modos de direccionamiento (MD1_out, MD2_out)
- Valor inmediato (Inmediato_out)

Integración en el Procesador:

- **Durante el Fetch**:
  - La instrucción se carga desde la memoria de código al IR (WIR='1').
- **Durante el Decode**:
  - Los campos del IR se distribuyen a:
  - **Unidad de Control**: S_out para generar señales de control.
  - **Unidad de Registros**: DA_out, DB_out, DC_out.
  - **Datapath**: Inmediato_out para operaciones con constante.
- **Ejecución**:
  - MD1_out y MD2_out controlan multiplexores en el Datapath.

3.6. Registro CW (Control Word)

Su función principal es gestionar y almacenar las señales de control que coordinan las operaciones del microprocesador, lo que quiere decir que almacena la palabra de control que es generada por la unidad de control.

Integración en el Procesador

1. **Durante el Decodificado**:
   - La Unidad de Control genera las señales de control basadas en el IR.

2. **Sincronización**:
    - Todas las señales se capturan en el Registro CW al activarse WCW.
3. **Durante la Ejecución**:
    - Las salidas del Registro CW controlan todos los componentes del datapath.

3.7. Decodificador de la Palabra de Control

Convierte la señal de Selección del RegistroIR en las señales de control que se almacenan en el RegistroCW, ya que transforma el código de una operación seleccionada en una palabra de control, que permite activar o desactivar señales específicas, también configura los modos de direccionamiento. Las salidas del decodificador se conectan a las entradas del RegistroCW.

Trabaja en conjunto con el RegistroIR porque dicho registro le proporciona el código de la operación y también alimenta al RegistroCW, todo este proceso permite implementar instrucciones a través de combinaciones de señales.

Utiliza 3 señales adicionales, las cuales fueron nombradas como A,B y C, las cuales provienen de un resultado hecho por ecuaciones booleanas utilizando la señal de selección.

Funciones booleanas de las 3 señales:

$$A <= Sin(4)' + Sin(4)Sin(3)'Sin(2)' + Sin(4)Sin(3)Sin(2)Sin(1)Sin(0)$$

$$B <= Sin(4)'Sin(2)Sin(1) + Sin(4)'Sin(2)'$$

$$C <= Sin(4)Sin(3)Sin(2)'Sin(1)Sin(0)'$$

3.9. Memoria de códigos

Almacena las instrucciones en número binario que el microprocesador debe ejecutar, también proporciona las instrucciones completas que luego son decodificadas y convertidas en señales de control, cada instrucción coincide con los campos de entrada del RegistroIR.

Trabaja en conjunto con los registros de control, para ejecutar cada instrucción de forma sincronizada.

3.9. Memoria de datos

La memoria de datos es un módulo de 256 bytes (8 bits de ancho x 256 direcciones).

La Memoria de Datos incluye una memoria de lectura y escritura. Esta memoria se usa para almacenar la información de la Unidad de Registros y también para recuperar datos de ella y cargarlos nuevamente en la Unidad de Registros cuando sea necesario.

Implementación Estructural

- Bancos de Memoria (8 módulos de 32 bytes)
    - 8 memorias de **32 bytes** cada una (Mem_Datos).
    - Cada banco maneja 5 bits de dirección (Dir[4:0]).
    - Los 3 bits superiores (Dir[7:5]) seleccionan el banco.
- Demultiplexor (Demux_1a8_8bits)
    - **Función**: Distribuye el dato de entrada (Datos_in) al banco seleccionado por Dir[7:5].
- **Multiplexor (Mux8a18Bits)**
    - **Función**: Selecciona la salida del banco activo según Dir[7:5].

3.10 Demultiplexor

El demultiplexor (Demux_1a8_8bits) dirige un vector de entrada de 8 bits (Valor_in) a una de sus 8 salidas (A-H), seleccionada por el código S[2:0]. Es clave en este sistema, donde se necesita distribuir datos a múltiples destinos, como en la memoria de datos.

Implementación Estructural

El módulo base es el Demux_1a8 cuya función es direccionar 1 bit a una de las 8 salidas según S[2:0]

El módulo Demux_1a8_8bits, está compuesto a partir de 8 instancias del componente Demux_1a8 (una por cada bit del bus).

Selección de banco: Dir[7:5] determina a qué banco (A-H) se escribe el dato.

Ecuaciones:

$a = (Valor\_in)(S(2)')(S(1)')(S(0)')$

$b = (Valor\_in)(S(2)')(S(1)')(S(0))$

$c = (Valor\_in)(S(2)')(S(1))(S(0)')$

$d = (Valor\_in)(S(2)')(S(1))(S(0))$

$e = (Valor\_in)(S(2))(S(1)')(S(0)')$

$f = (Valor\_in)(S(2))(S(1)')(S(0))$

$g = (Valor\_in)(S(2))(S(1))(S(0)')$

$h = (Valor\_in)(S(2))(S(1))(S(0))$

3.11. Registro Banderas

El registro de banderas almacena temporalmente los indicadores de estado del procesador (Carry, Overflow, Signo, Zero) generados por la ALU, permitiendo su uso posterior en saltos condicionales y otras operaciones de control.

- **Entradas**:
    - C_in, V_in, S_in, Z_in: Banderas desde la ALU.
    - WE: Habilita escritura (activo en alto).
    - clk: Sincronización (flanco ascendente).
- **Salidas**:
    - C_out, V_out, S_out, Z_out: Banderas almacenadas.

3.12. Unidad de registros

La Unidad de Registros (UR) es un banco de 8 registros de 8 bits que permite:

- Escritura de datos en un registro específico.
- Lectura simultánea de dos registros para operaciones con dos operandos.
- Control mediante señales de selección y habilitación.

Implementación Estructural:

La unidad de registros está construida a partir de dos módulos.

1. **Módulo UR_In (Escritura)**

   Gestiona solo la escritura en registros.

- **Entradas**:
    - C[7:0]: Dato a escribir.
    - Dc[2:0]: Selección del registro destino (0-7).
    - W: Habilitación de escritura (activo alto).
    - clk: Sincronización (flanco ascendente).
- **Componentes**:
    - Decodificador 3:8
    - Compuertas AND
    - Registros de 8 bits

2. **Módulo UR_Complete (Lectura/Escritura)**

   Combina UR_In con multiplexores para lectura/escritura completa, formando la unidad de registros funcional

- **Entradas:**
  - Da[2:0], Db[2:0]: Selección de registros para lectura en buses Ain y Bin.
  - CIN, Dc, W, clk: Igual que UR_In.
- **Componentes**:
  - Instancia de UR_In
  - Multiplexores 8:1

Integración en el Procesador:

- **Desde la ALU**:
  - CIN recibe resultados de operaciones (ej: ADD R1, R2 → resultado a R1).
- **Hacia la ALU**:
  - Ain y Bin proporcionan operandos (ej: R1 y R2 para ADD).
- **Control**:
  - La Unidad de Control (UC) genera Da, Db, Dc, y W según la instrucción.

3.13. Unidad de desplazamiento y rotación

El módulo Shifter está implementado con multiplexores y realiza desplazamientos y rotaciones de 8 bits, hace 4 diferentes operaciones controladas por señales de selección (S[1:0]). Genera salidas de carry para operaciones extendidas.

Operaciones que realiza:

S[1:0]	**Operación**	**Descripción**	**Carry Izq. (Cl) = bit más significativo original**	**Carry Der. (CD) = bit menos significativo original**
00	SHR	Desplaza a la derecha (entra '0')	-	Ain[0]

01	SHL	Desplaza a la izquierda (entra '0')	Ain[7]	-
10	ROR	Rotación circular a derecha	-	Ain[0]
11	ROL	Rotación circular a izquierda	Ain[7]	-

Tabla .3.3.4. Unidad de desplazamiento y rotación.

Implementación Estructural:

El diseño de la Unidad de Desplazamiento y Rotación consiste en seleccionar los bits adecuados mediante multiplexores.

1. **Multiplexores 4:1 (Mux4a1)**:
    - **Función**: Cada bit de salida R[i] se selecciona entre 4 posibles fuentes, controladas por S[1:0].
    - **Implementación**:
        - 8 multiplexores (uno por bit) conectados en paralelo.
        - Cada Mux4a1 elige entre:
            - Bit desplazado/rotado desde la izquierda/derecha.
            - Bit original con nueva posición.
2. **Lógica de Carry**:
    - Cl (Carry Left): Captura el MSB (Ain[7]) antes del desplazamiento.
    - Cr (Carry Right): Captura el LSB (Ain[0]) antes del desplazamiento.

3.14. Secuenciador

Es el componente encargado de generar las señales de control necesarias para la operación de una unidad de procesamiento, basado en un estado actual y condiciones de entrada.

Lógica de Operación:

- **Entradas**:
    - J: Indica si la instrucción actual es un salto
    - Sv: Indica si se cumple la condición de salto
    - clk: Señal de reloj para la sincronización
- **Salidas**:
    - WIR : Escribir en el registro de instrucción
    - WCW : Escribir palabra de control
    - MB:Selección del multiplexor B
    - MC:Selección del multiplexor C
    - WPC: Escribir en el contador de programa

Operaciones que realiza:

Estado	Señales Activas	Operación Realizada	Descripción
000	Ini	Inicialización/reset	Prepara el sistema para comenzar la ejecución
001	I	Carga de instrucción (Fetch)	Obtener la siguiente instrucción desde memoria
010	A	Decodificación/escritura en la palabra de control	Interpreta la instrucción y configura las señales de control
011	B	Evaluación de salto (J)	Decide si se toma o no un salto condicional
100	C	Salto no tomado (PC +1)	Continúa con la siguiente instrucción cuando no se cumple la condición de salto
101	D	Salto tomado (MB=1)	Desvía el flujo de ejecución cuando se cumple la condición de salto.
110	E	Operación especial	Maneja casos excepcionales o instrucciones especiales
111	01	—	—

Tabla .3.3.5. Secuenciador.

Tabla de estados:

F2	F1	F0	J	SV	F2+	F1+	F0+
0	0	0	X	X	0	0	1
0	0	1	X	X	0	1	0
0	1	0	X	X	0	1	1
0	1	1	0	X	1	0	0
0	1	1	1	0	1	1	0
0	1	1	1	1	1	0	1
1	0	0	X	X	X	X	X
1	1	1	X	X	X	X	X
1	0	0	X	X	X	X	X

Tabla 3.3.6. Estados del Secuenciador

Ecuaciones:

D2= F2’F1F0J’ + F2’ F1F0JSV’ + F2’F1F0JSV

D1= F2’F1’F0 + F2’F10’ + F2’F1F0JSV’

D0= F2’F1’F0’ + F2’F1F0’ + F2’F1F0JSV

Carta ASM:

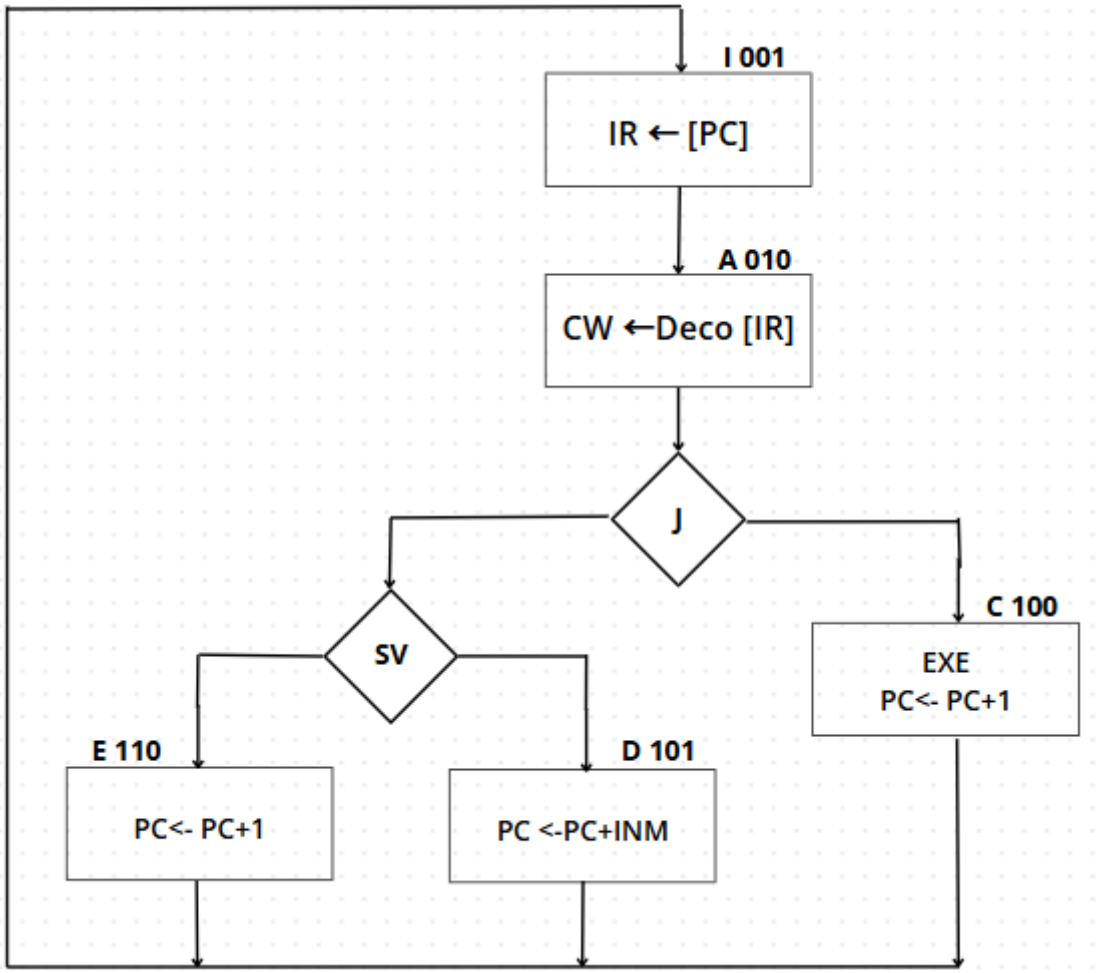


Figura 3.14.1.Carta ASM.

3.15. Registro PC

La unidad de Contador de Programa (PC) es la responsable de gestionar la dirección de la siguiente instrucción que se va a ejecutar, esto permite que existan secuencias lineales (PC+1) o saltos condicionales, a través de señales de control.

- **Entradas**:
    - Inmediato: Valor de dirección para saltos, proviene de la instrucción
    - MB: Selección del multiplexor B, controla si se usa un inmediato o “00000000”)
    - MC: Selección del multiplexor C, controla si se suma 1 o 0 al PC.
    - WPC: Actualiza el registro del PC.
    - clk: Señal de reloj
- **Salidas**:

- COPC : Acarreo de salida del sumador, indica si hay desbordamiento
- PC : Valor actual del Contador de Programa (PC)

Tabla de Funcionamiento

Señales de Control	Operación Realizada	Fórmula	Descripción
MB= 0, MC= 1	Incremento normal (PC+1)	PC <- PC+1	Avanza secuencialmente a la siguiente instrucción en memoria
MB= 1, MC= 0	Salto con carga de Inmediato	PC<- Inmediato	Salta a una dirección especificada con el inmediato de la instrucción
MB= 1, MC= 1	Salto: PC + Inmediato +1	PC <- PC+ Inmediato +1	Salta a una dirección relativa al PC actual
WPC= 0	Mantener valor actual	PC no cambia	Congela el PC actual

Tabla 3.3.77. Registro PC.

3.16. Unidad Funcional (UF)

Es el núcleo de ejecución de operaciones aritméticas, lógicas y de desplazamiento en el microprocesador. Combina una ALU junto con un Shifter, esto genera resultados y banderas de estado a través de un control unificado.

Lógica de Operación

- **Entradas**:
  - A: Operando de 8 bits
  - B: Operando de 8 bits.
  - S: Selector de operación, es de 5 bits de los cuales 4 bits son para la ALU y 1 bit selecciona entre la ALU o el Shifter
  - carry_in: Acarreo de entrada
- **Salidas**:
  - Cout : Acarreo de salida
  - V : Bandera de Overflow
  - Z: Bandera de cero
  - Signo: Es el bit más significativo del resultado
  - Rout: Resultado de la operación, es de 8 bits

Implementación Estructural:

La unidad funcional está construida a partir de los siguientes módulos.

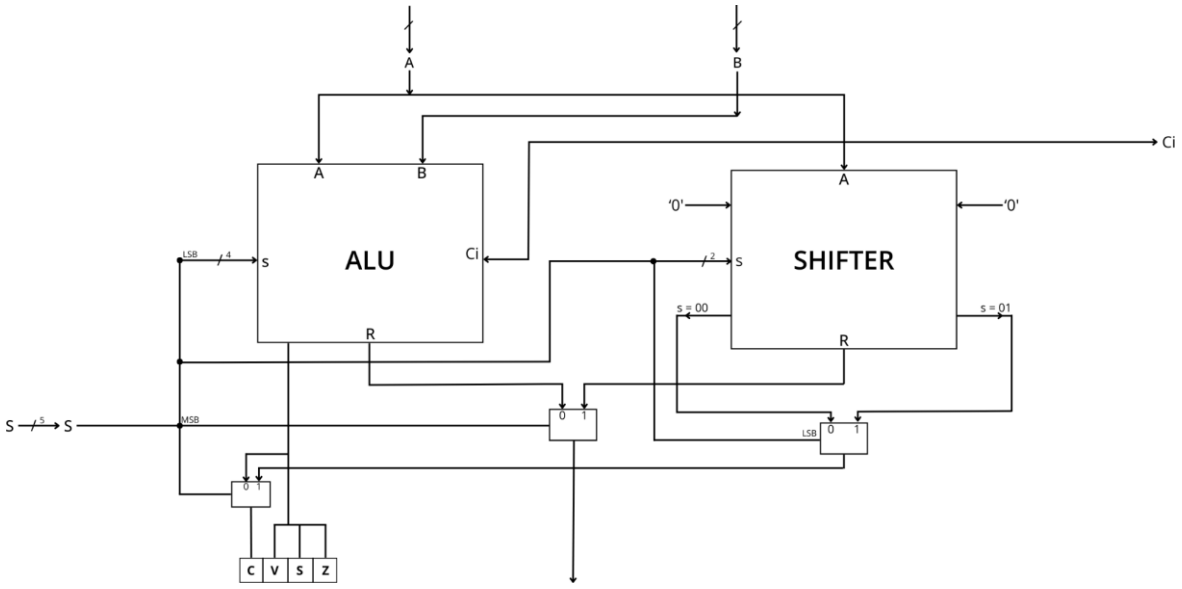


Figura 3.16.1. Diagrama de bloques de la Unidad Funcional

1) ALU (Unidad Aritmética-Lógica)

Las operaciones se controlan por S, se pueden realizar 16 operaciones.

- **Banderas generadas**:
  - Correo: Acarreo en sumas o restas.
  - Vover: Overflow en operaciones aritméticas.
  - Ze: Resultado cero.
  - Sig: Es el bit más significativo del resultado.

2)Shifter

Las operaciones se controlan por S, es posible ejecutar 4 modos, dichos modos son:

  - 00: Desplazamiento izquierdo
  - 01: Desplazamiento derecho.
  - 10: Rotación a la izquierda.
  - 11: Rotación a la derecha.

- **Salidas**:
  - Cl: Acarreo utilizado en el desplazamiento a la izquierda.

- Cr: Acarreo utilizado en el desplazamiento a la derecha.

3) Multiplexores

- Mux_2_a_1_C: Utilizado para seleccionar entre el acarreo generado por la rotación a la izquierda o el acarreo generado por la rotación a la derecha.
- Mux_2_a1_Co: Utilizado para seleccionar entre el acarreo generado por la ALU o el acarreo generado por el Shifter.
- Mux_2_a_1_UF: Utilizado para seleccionar el resultado R1 (resultado de la ALU) o R2(resultado del Shifter).

Tabla de operaciones:

Selector	Operación	Componente utilizado	Efecto en banderas
0	Operación en ALU	ALU	Cout, V, Z y Signo
1	Desplazamiento	Shifter	Cout

Tabla 3.3.8. Selecciones.

3.17. Datapath

Es el componente principal del microprocesador, el cual integra todos los elementos necesarios para ejecutar instrucciones, dichas instrucciones son:

- Operaciones aritméticas
- Acceso a memoria
- Manejo de registros
- Evaluación de saltos condicionales

Lógica de Operación

- **Entradas**:
    - Inmediato: Valor constante utilizado para operaciones o saltos
    - Da: Dirección de registro, tanto de lectura como escritura.
    - Db: Dirección de registro, tanto de lectura como escritura.
    - Dc:Dirección de registro, tanto de lectura como escritura.

- **Señales de control**:
    - W: Utilizado para habilitar la escritura en los registros
    - WE: Utilizado para habilitar la escritura en el registro de banderas
    - MI: Se utiliza para seleccionar el operando, dichos operandos son registro o inmediato
    - MM:Selección de resultado, tanto para la ALU como para la memoria.
    - MW: Utilizado para habilitar la escritura en la memoria de datos
    - Selector: Controla la operación que se va a ejecutar en la UF (Unidad Funcional)
    - clk: Señal de reloj
- **Salidas**:
    - Banderas:
        - Carry_out, Vo_out, Zero_out, Signo_out: Representan el estado actual de las banderas
    - Control:
        - CMP:Es el resultado de comparación
        - SV: Indica si el salto es válido, para ello es necesario cumplirse la condición de salto
        - Result: Resultado de la operación que se ejecutó, es de 8 bits.

Implementación Estructural:

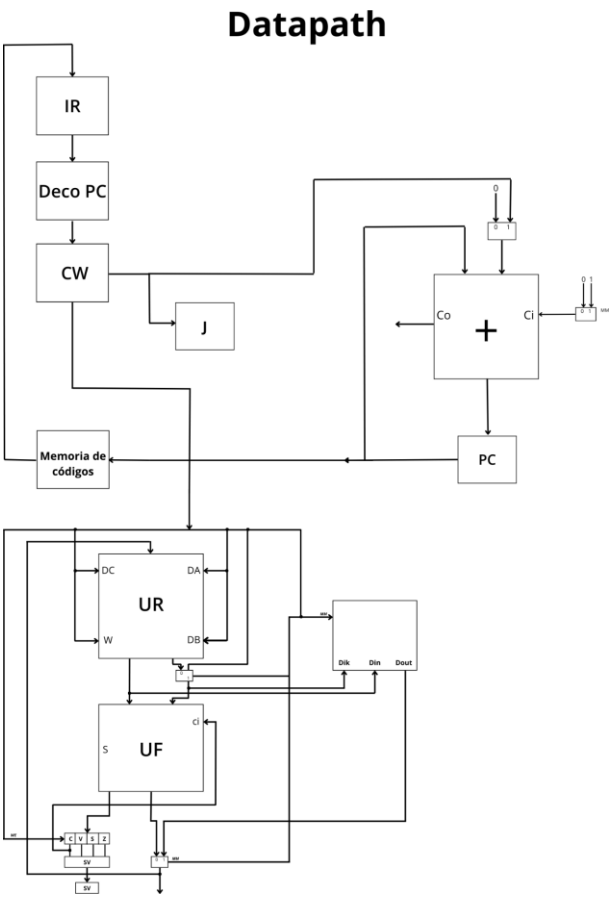


Figura 3.17.1. Diagrama de bloques del Datapath

1) Unidad de Registros (UR_Complete)

Almacena y recupera datos en los registros.

- **Entradas**:
  - CIN: Dato que se va a escribir, proviene de la ALU o de la memoria
  - Da,B: Registros que se van a leer
  - Dc: Registro a escribir cuando W=1
- **Salidas**:
  - Ain, Bin: Valores que se leyeron de los registros

2) Unidad Funcional (UF)

Lleva a cabo la ejecución de operaciones aritméticas y/o lógicas, también operaciones de desplazamiento.

- **Banderas generadas**:
    - Cout
    - V
    - Z
    - Signo

  Se actualizan si WE=1.

3) Memoria de Datos (Mem_Datos_256R)

Tiene una capacidad de almacenamiento de 256 bytes.

- **Acceso a la memoria**:

  Escribe A_UnidadRegistros en la Dir si MW=1 y siempre lee en la dirección Dir.

4) Comparador (Comparador)

Comparando A y B genera un resultado CMP=1 en el caso de que exista una igualdad, sino A y B no son iguales.

5) Multiplexores

- Mux_MI: Selecciona entre B_UnidadRegistros (es un registro) o Inmediato (es una constante)
- Mux_MM: Selecciona entre Resultado_UF(ALU o Shifter) o Mem_Datos_Out (dato que se leyó en memoria)

6) Registro de Banderas (Registro_banderas)

Almacena las banderas C,V,Z,Signo cuando WE=1.

7) Unidad de Salto Válido (Salto_Valido)

Verifica si se cumple una condición de salto, se basa en las banderas y en el Selector.

Tabla de funcionamiento:

Señal de Control	Operación	Ruta de Datos
MI=0	Utiliza un registro (B_UnidadRegistros)	Mux_MI = B_UnidadRegistros
MI=1	Utiliza un inmediato(Inmediato)	Mux_MI = Inmediato
MM=0	Resultado proveniente de la ALU (Resultado_UF)	Mux_MM = Resultado_UF (Escribe en registro)
MM=1	Resultado desde una memoria	Mux_MM = Mem_Datos_Out
MW=1	Escritura en una memoria	Mem_Datos_256R almacena A_UnidadRegistros
W=1	Escritura en un registro	UR_Complete guarda Mux_MM en Dc

Tabla 3.3.9. Funcionamiento Datapath.

Flujo de Datos:

1) Fetch / Decodificación
   Los registros A y B se leen (Da, Db), después el MUX_MI selecciona el operando B o el Inmediato.
2) Ejecución
   La UF opera con A y Mux_MI, esto genera Resultado_UF y banderas.
3) Escritura
   Mux_MM decide si el resultado proviene de la ALU o memoria, si W=1 se escribe en el registro Dc.
4) Banderas / Saltos
   Las banderas se actualizan cuando WE=1, el Salto_Valido verifica si se cumple la condición de salto (SV).

4. Implementación del Microprocesador

4.1 Desafíos

El principal problema fue lograr una correcta sincronización entre la gran cantidad de módulos ya que no debe haber ninguna pérdida de bits, por lo tanto fue un gran reto asegurarse que todo estuviera sincronizado. Por lo que se tuvo que llevar a cabo una organización en la palabra de control, gracias a ello fue posible obtener dicha sincronización.

4.1 Instrucciones de transferencia de datos

Mueven un dato de un lugar del microprocesador a otro sin cambiar el dato. Las transferencias típicas son entre la memoria y los registros del procesador.

- La instrucción Load se utiliza para indicar una transferencia desde la memoria a un registro del microprocesador.
- La instrucción Store indica una transferencia desde un registro del microprocesador a una posición de memoria.
- La instrucción Move se utiliza en microprocesadores con múltiples registros para indicar una transferencia desde un registro del microprocesador a otro registro.

Nombre	Mnemónico
Load	LD
Store	ST
Move	MOVE

Figura 4.1. Instrucciones de transferencia de datos.

4.2 Instrucciones aritméticas

Permiten realizar cálculos matemáticos básicos utilizando datos binarios que están almacenados en registros o en memorias.

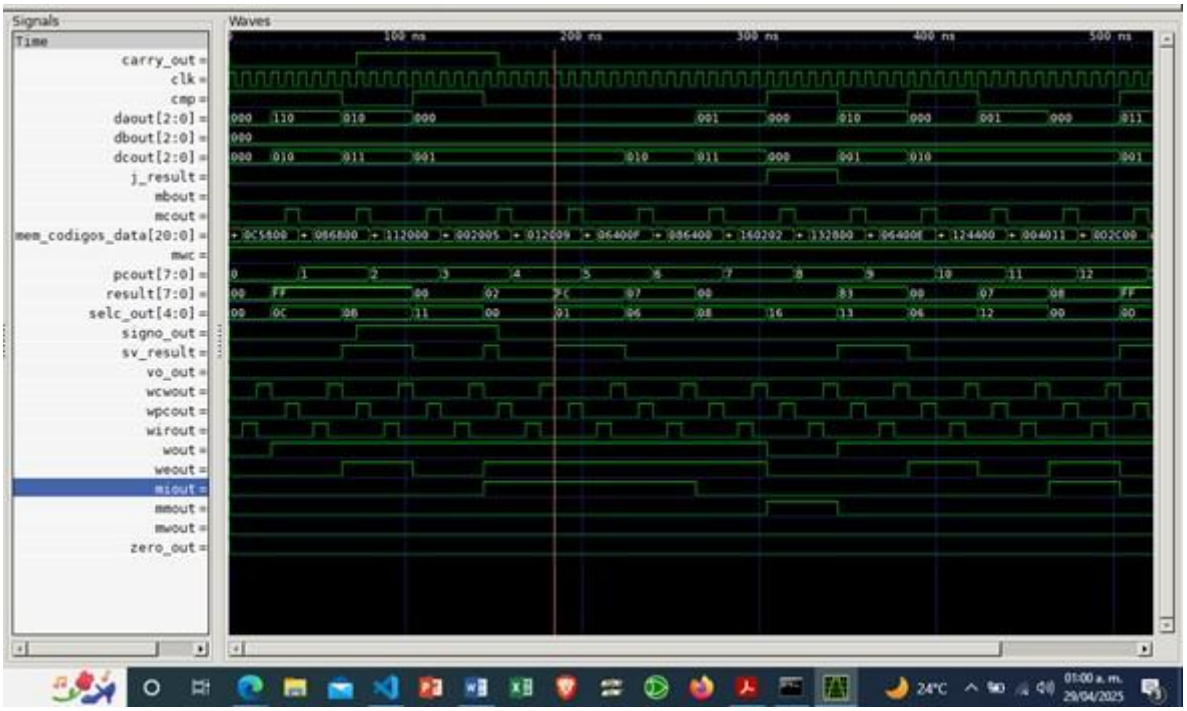
- La instrucción de incremento suma uno al valor almacenado en un registro de la memoria o palabra de la memoria.
- La instrucción de decremento resta uno al valor almacenado en un registro del microprocesador o también de una posición de memoria.
- Las instrucciones de suma, resta, multiplicación y división están disponibles para diferentes operaciones con datos.
- La instrucción suma con acarreo ejecuta una operación de dos operandos y también suma el valor del acarreo del cálculo anterior.

Nombre	Mnemónico
Incremento	INC
Decremento	DEC
Suma	ADD
Resta	SUB
Multiplicación	MUL
División	DIV
Suma con acarreo	ADDC

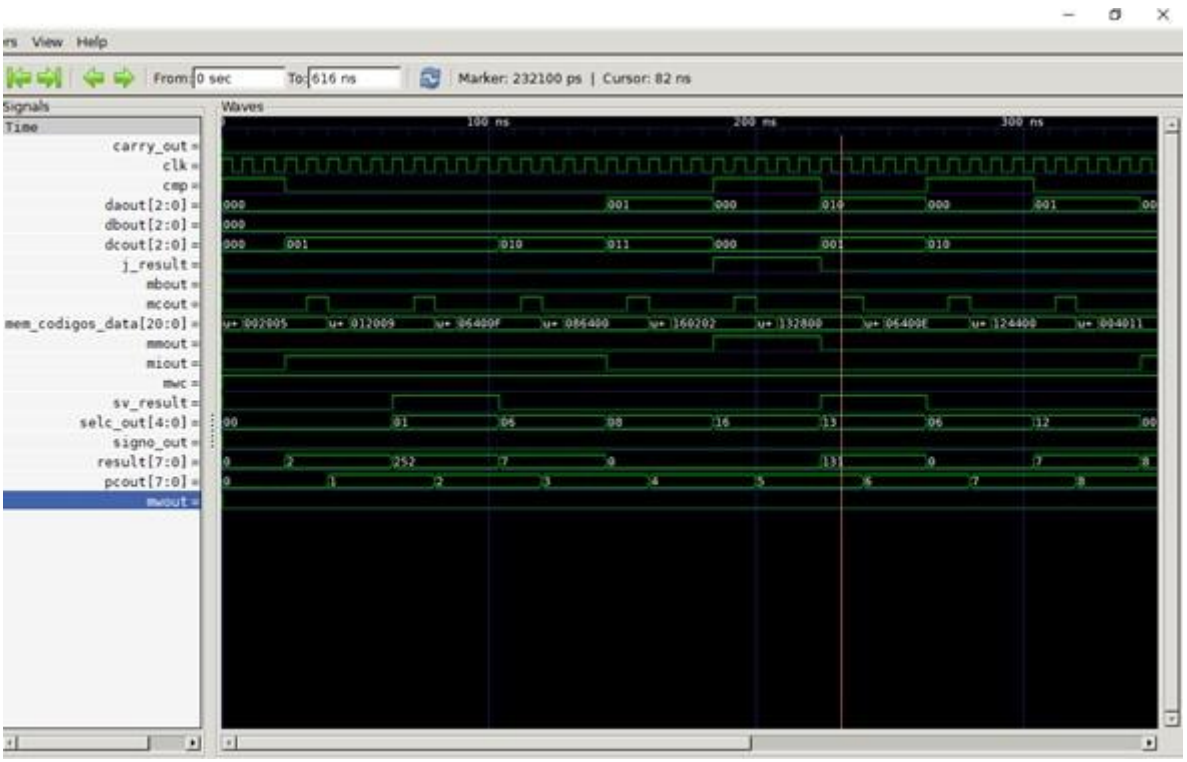
Figura 4.2. Instrucciones aritméticas.

5. Pruebas y Resultados del Prototipo

Prueba 1. Funcionamiento del microprocesador.



Prueba 2. Funcionamiento del microprocesador.



Prueba 3. Funcionamiento del microprocesador.

6. Conclusiones

Se cumplieron todos los objetivos de la implementación del microprocesador, los cuales se obtuvieron al momento de realizar múltiples pruebas utilizando diferentes operaciones y valores agregados a través de un inmediato, dichas operaciones se especificaron en la palabra de control. Se observaron los resultados y comportamientos que genera cada una de las operaciones o instrucciones.

Para lograr los resultados fue necesario implementar todos los aprendizajes obtenidos referente a la arquitectura de computadoras.

La implementación del microprocesador se logró mediante el uso de distintos componentes previamente realizados, después se fueron creando bloques completos que estaban construidos con los bloques más pequeños.