Diseño de un microprocesador

Maripura Sánchez Méndez
Blanca Concepción Alvarez
Candidatas a graduación

Sinopsis

Presentamos el diseño de un microprocesador que incluye el conjunto de instrucciones, los modos de dirección y el ciclo de leer-decodificar-ejecutar una instrucción. Los registros usados para el diseño se escogieron de la tarjeta programable y se definieron sus funciones.

Abstract

We show the design of a microprocessor that includes the set of instructions, the addressing modes and the cycle of fetch-decode-execute an instruction. We chose the registers used in the design from the board and then defined their functions.

Introducción

En la actualidad los microprocesadores se usan para controlar el funcionamiento de muchos sistemas. Hay varias clases de microprocesadores en el mercado, todos con características distintas. Para seleccionar un microprocesador que controle un sistema particular hay que estudiar el trabajo que realiza el sistema y determinar cuáles de los microprocesadores disponibles en el mercado cumplen con los requisitos para realizar el trabajo. Luego se selecciona el microprocesador ideal para ese sistema.

Para entender mejor cómo es que los microprocesadores logran realizar estos trabajos, diseñamos un microprocesador. Comenzamos definiendo la arquitectura del microprocesador, luego diseñamos la microprogramación de las instrucciones y, por último, realizamos la fase de pruebas para verificar que la unidad central de procesamiento (CPU) funcionara correctamente.

Investigación

Nuestro proyecto comienza investigando y estudiando los conceptos necesarios para diseñar un microprocesador. Luego de leer varios libros y estudiar las características de diferentes microprocesadores, establecimos un procedimiento de diseño. Este procedimiento consiste de los siguientes pasos:

- Definir las características del microprocesador
- Escoger y definir el conjunto de instrucciones
- Diseñar la arquitectura del microprocesador
- Diseñar el microprograma
- Probar el sistema

Diseño

Especificaciones del diseño

Al definir las características del microprocesador, primeramente establecimos que el diseño fuera sencillo, esto es, que ejecutara las funciones mínimás que un microprocesador debe tener. Aunque originalmente decidimos que el tamaño del microprocesador sería de ocho dígitos binarios, luego tuvimos que aumentarlo a 16 dígitos binarios por razones que se explican más adelante. Entre las opciones de los diferentes tamaños de microprocesadores de hoy día, consideramos que ocho dígitos binarios era un tamaño adecuado para nuestro primer diseño.

Usamos el método de dirección sencilla para realizar las operaciones. Esta forma de dirección ejecuta todas las operaciones usando siempre el acumulador como el primer operando. Para obtener el segundo operando (si la instrucción lo necesita) el CPU realiza una serie de microinstrucciones para localizarlo, leerlo y guardarlo. Si se hubiese usado el método de dirección doble, entonces el CPU tendría que obtener los dos operandos para luego ejecutar la operación. Estos pasos alargan aún más el proceso de ejecutar una instrucción y complican la microprogramación.

Selección del conjunto de instrucciones

Según las especificaciones del diseño, escogimos un conjunto de instrucciones que realiza funciones básicas:

- ADD y SUBS
  Se necesita una instrucción que sume (ADD) y otra que reste (SUBS).
- AND, OR y XOR
  Para realizar las operaciones lógicas escogimos las instrucciones AND, OR y XOR. Para obtener las demás operaciones lógicas se pueden combinar estas tres operaciones.
- JP, JPS, BNZ, RT y CMP
  Todo programa debe tener la opción de hacer lazos ("loops") y usar subrutinas para ejecutar una función. Por esta razón incluimos las instrucciones de "Jump" (JP) y "Jump Subroutine" (JPS). Luego, cuando definimos las instrucciones, notamos que necesitábamos una instrucción para ramificar y añadimos la instrucción "Branch" (BNZ). También añadimos instrucciones para indicar cuándo continuar la secuencia orignal del programa una vez se ejecuta la subrutina o decidir cuándo terminar la ramificación ["Return" (RT) y "Compare" (CMP)].
- MUL y DIV
  Para las operaciones de multiplicación y división añadimos las instrucciones MUL y DIV. Estas instrucciones sólo se pueden

ejecutar con números de punto flotante.

- CMT y NEG

  Para complementar los números escogimos la instrucción "Complement" (CMT) y para cambiar el signo de los números escogimos la instrucción "Negate" (NEG).

- RTL, RTR, SL y SR

  Para hacer operaciones de multiplicación con números binarios escogimos las siguientes instrucciones: "Rotate Left" (RTL), "Rotate Right" (RTR), "Shift Left" (SL), "Shift Right" (SR).

- STP

  Escogimos una operación para detener la ejecución de un programa: "Stop" (STP).

- NOP

  Añadimos una instrucción que no ejecuta ninguna operación. Si el programa necesita tiempo para ejecutar una operación antes de leer la próxima instrucción, entonces usamos la instrucción "No Operation" (NOP).

- Luego de realizado el microprograma necesitamos una instrucción de suma y otro de resta para números de punto flotante y las incluimos en el conjunto. Las razones se explican más adelante.

Para seleccionar los modos de dirección especificamos que el microprocesador sólo obtendría información directamente de la instrucción, de los registros y de la memoria. Entonces definimos los siguientes modos de dirección: inmediato (de la instrucción), directo (de la memoria), de registros (de los registros). En algunos casos hay instrucciones que no necesitan un segundo operando o ningún operando para realizar una operación. A esta forma de dirección le llamamos el modo implícito, ya que sólo se ejecuta con la instrucción y no necesita otra información.

Luego de tener el conjunto de instrucciones y los modos de dirección, establecimos el formato de la instrucción. Cuando comenzamos a agrupar los campos nos dimos cuenta que no era suficiente ocho dígitos binarios para ejecutar las instrucciones y aumentamos a 16 dígitos binarios. De esta

manera acomodamos toda la información requerida para ejecutar una instrucción.

Diseño de la arquitectura

Diseñamos la arquitectura del microprocesador de la siguiente manera:

- Contador de programa (PC)
  Se requiere un contador de programa para llevar la secuencia del programa.
- Registro acumulador (ACC)
  El registro acumulador tiene siempre el valor del primer operando y almacena el resultado luego de ejecutarse la operación, destruyendo la información que tenía anteriormente. Necesitamos este registro porque nuestro diseño usa el modo de dirección sencillo.
- Registro de datos de memoria (MAR)
  El registro de datos de memoria almacena el operando que se lee de la memoria principal o que se va a escribir en la memoria. Cuando se usa el modo de dirección directo este registro guarda el segundo operando.
- Registro de datos (DR)
  El registro de datos almacena el segundo operando que se va a usar en una operación cuando se usan los modos de dirección inmediato y de registro.
- Registro de pila (SP)
  El conjunto de instrucciones tenía algunas instrucciones de brinco a subrutinas. Entonces necesitábamos un registro para almacenar la localización de memoria donde ocurrió el brinco para luego continuar la secuencia normal del programa. Para este propósito añadimos el registro de pila.
- Registro de memoria de dirección (MAR)
  El registro de memoria de dirección indica la localización de la memoria para la próxima lectura o escritura.

- Registro de instrucción (IR)
  El registro de instrucción almacena la instrucción que se estaba ejecutando para luego decodificarla.
- Registros de propósito general (RB y RC)
  Aunque nuestro microprocesador usa el modo de dirección sencilla, decidimos añadir dos registros de propósito general. De esta manera podemos almacenar información en el CPU sin tener que usar las localizaciones de memoria, proceso que tomaría más tiempo. A base de las especificaciones de nuestro diseño sencillo escogimos dos registros (RB y RC).
- Registro de banderas
  El registro de banderas indica el estado de una operación para la instrucción de brinco con condición y señala cuándo el resultado de la operación "Compare" sea cero
- Mientras desarrollábamos la microprogramación añadimos tres registros que se usan para almacenar la información a decodificarse.

Diseño del microprograma

Antes de comenzar a diseñar el microprograma teníamos que aprender a usar la tarjeta microprogramable. Comenzamos estudiando el manual de la tarjeta y analizando los ejemplos de microprogramación incluidos. Estudiamos el formato de las microinstrucciones y luego aprendimos a usar el "software" de la tarjeta. Entonces comenzamos a diseñar nuestro microprograma.

Primeramente diseñamos la tabla de definiciones para el ensamblador y luego el microcódigo. El primer microcódigo que diseñamos fue de dos instrucciones. Para observar cómo funcionaba el microcódigo, hicimos un programa en lenguaje de ensamblaje ("assembly language") con las mismas instrucciones que estaban en el microcódigo. Según añadíamos una instrucción, corríamos el programa para ver cómo funcionaba, corregíamos los errores y continuábamos añadiendo instrucciones al microcódigo. Cuando terminamos con todas las instrucciones del microcódigo, también

teníamos un programa en lenguaje de ensamblaje que probaba todas las instrucciones.

Fase de prueba

Aunque ya teníamos un programa que probaba todas las instrucciones, decidimos hacer otro que realizara una función. Sin embargo, como el nuevo programa iba a usar números de punto flotante para simular la función, tuvimos que añadir dos instrucciones a nuestro conjunto de instrucciones. Hasta este punto era posible multiplicar y dividir correctamente, pero no se podía sumar ni restar números de punto flotante. Para este propósito añadimos las instrucciones "DADD" Y "DSUB".

Equipo usado

Realizamos la emulación del CPU en el "TI 8800 Software Development Board (SDB)". Este sistema consiste de una tarjeta microprogramable de 32 dígitos binarios que se instala en una computadora IBM PC/AT o compatible. La tarjeta tiene implantado un control de almacenaje ("control store" - localización de almacenaje que contiene el microcódigo) de escritura en el "Random Access Memory" (RAM por sus siglas en inglés).

El microprocesador 8800 SDB consiste de los siguientes componentes:

- memoria de microcódigo
- microsecuenciador
- unidad de lógica aritmética (ALU) de 32 dígitos binarios
- procesador de punto flotante y enteros
- memoria local de datos

La figura 1 presenta el diagrama de bloques del microprocesador TI 8800.

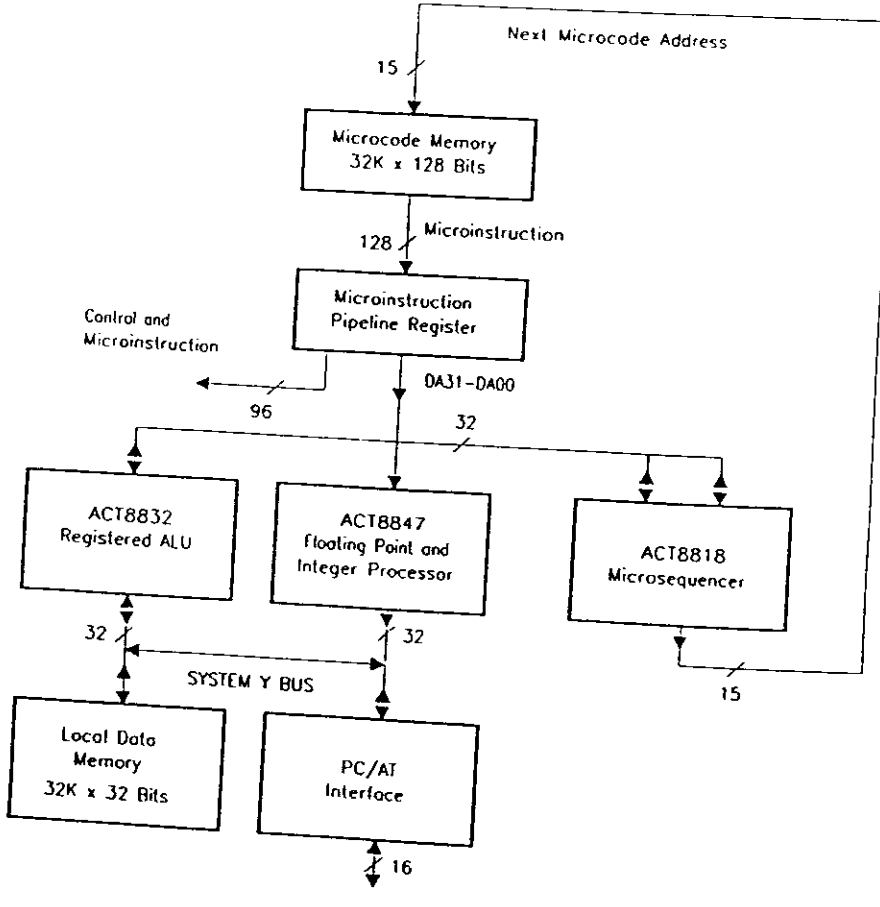

Figura 1: Diagrama de bloques del microprocesador TI 8800.

Dos líneas de transmisión conectan los componentes internos del sistema. La línea de transmisión "DA" provee datos desde el campo de datos de la microinstrucción al ALU, al procesador de punto flotante o al microsecuenciador. Esta línea también puede usarla el ALU o el microsecuenciador para proveerles datos a otros componentes del sistema. La otra línea de transmisión es la línea Y, que conecta el ALU y el procesador de punto flotante a la memoria local.

Microsecuenciador

La función principal del microsecuenciador es producir la dirección de la próxima microinstrucción a ejecutarse. Esta dirección de 15 dígitos binarios se obtiene de la memoria de microcódigo (memoria que almacena el microcódigo).

ALU con registros

El ALU consiste de 64 registros de 32 dígitos binarios que se pueden configurar para operar como cuatro registros de ocho dígitos binarios, dos registros de 16 dígitos binarios o uno de 32 dígitos binarios. El procesador incluye 13 funciones aritméticas y lógicas con ocho desplazadores ("shifts") condicionales, multiplicación, división, suma, resta y conversión de datos.

Procesador de punto flotante y enteros

El "floating point unit" realiza operaciones aritméticas y lógicas enteras y desplazadores lógicos, conversión de valores absolutos, conversiones de punto flotante a entero y viceversa.

Formato de la microinstrucción

El formato de la microinstrucción del 8800 consiste de 128 dígitos binarios divididos en 30 campos funcionales (tabla 1).

Tabla 1 - Valores por falta para los campos

Field	# of bits	Description
		Control of board
1	5	Select condition code input
2	1	Enable/disable external I/O request signal
3	2	Enable/disable local data memory read/write operations
4	1	Load status/do not load status
5	2	Determine unit driving Y bus
6	2	Determine unit driving DA bus
		8847 floating point and integer processing chip
7	1	C register control: clock, do not clock
8	1	Select most significant or least significant bits for Y bus
9	1	C register data source: ALU, multiplexer
10	4	Select IEEE or FAST mode for ALU and MUL
11	8	Select sources for data operands: RA registers, RB registers, P register, S register, C register
12	1	RB register control: clock, do not clock
13	1	RA register control: clock, do not clock
14	2	Data source confirmation
15	2	Enable/disable pipeline registers
16	11	8847 ALU function
		8832 Registered ALU
17	2	Write enable/disable data output to selected register: most significant half, least significant half
18	2	Select register file data source: DA bus, DB bus, ALU Y MUX output, system Y bus
19	3	Shift instruction modifier
20	1	Carry in: force, do not force
21	2	Set ALU configuration mode: 32, 16 or 8 bits
22	2	Select input to S multiplexer: register file, DB bus, MQ register
23	1	Select input to R multiplexer: register file, DA bus
24	6	Select register in file C for write
25	6	Select register in file B for read
26	6	Select register in file A for write
27	8	ALU function
		8878 Microsequencer
28	12	Control input signals to the 8818
		WCS data field
29	16	Most significant bits of writeable control store data field
30	16	Least significant bits of writeable control store data field

Cada campo consiste de uno o más dígitos binarios y están agrupados en cinco categorías primordiales:

- Tarjeta de control
- Procesador de punto flotante y enteros 8847
- ALU con registros 8832
- Microsecuenciador 8818
- Campo de datos de control de almacenaje de escritura ("Writeable control store" - WCS, por sus siglas en inglés)

Información del "software" para el emulador TI 8800 SDB

El microprograma consiste de dos programas: el ensamblador y el microcódigo. En esta sección se describen los pasos a seguir para desarrollar un microprograma en el emulador:

1. Escribir la tabla de definiciones para el ensamblador.

   Este programa interpreta las instrucciones en lenguaje de ensamblaje y las traduce a instrucciones de microprogramación. Se puede usar cualquier editor ASCII para escribir el código. Le asignamos el nombre "**DEFINI.SRC**". La extensión siempre es SRC.

2. Compilar el ensamblador

   Para compilar el ensamblador se ejecuta instrucción **TIM DEFINI**. Esta instrucción convierte los códigos simbólicos de un programa a un código de lenguaje de máquina y le asigna una dirección de memoria a cada instrucción. Hay que corregir los errores hasta que el programa compile correctamente.

4. Escribir el microcódigo

   El microcódigo es un programa que define las funciones del

microprocesador y ejecuta las operaciones. Se le asigna la extensión SRC (**CODIGO.SRC**). Para compilar el microcódigo se ejecuta la instrucción **MICRO CODIGO**. Esta instrucción ensambla y carga el programa en el WCS.

5. Escribir programa de prueba

El programa de prueba se escribe en lenguaje de ensamblaje y se usa para comprobar que el diseño funciona correctamente. Se puede hacer un programa que pruebe las instrucciones o que ejecute una función que compruebe que el diseño funciona. Al programa se le asigna la extensión SRC (**PRUEBA.SRC**). Para ensamblar el programa se ejecuta la instrucción **ASM PRUEBA DEFINI**. Esta instrucción también carga el programa en la tarjeta.

6. Correr el programa

El programa de interface SDB le permite al usuario examinar cómo funciona el microprograma en el "TI 8800 SDB". Para ejecutar este programa se escribe la instrucción **SDB**. En el menú que se presenta en la pantalla se selecciona la opción de "Single step", la cual ejecuta el programa paso a paso. Esta opción nos lleva a otra pantalla (fig. 2) en la que se pueden observar los registros del "SDB" y la microinstrucción que se está ejecutando. Esta pantalla se divide en cuatro secciones: 8832 ALU, 8847 unidad de punto flotante, 8818 secuenciador y el estado de la tarjeta SDB. En la parte inferior de la pantalla se puede observar la línea de comandos, por la cual el usuario entra los comandos que desea ejecutar. La parte superior de la pantalla muestra el contenido actual de los registros. El registro afectado o que la microinstrucción que se ejecuta va a usar se ilumina. El campo de la instrucción se muestra usando nombres mnemónicos.

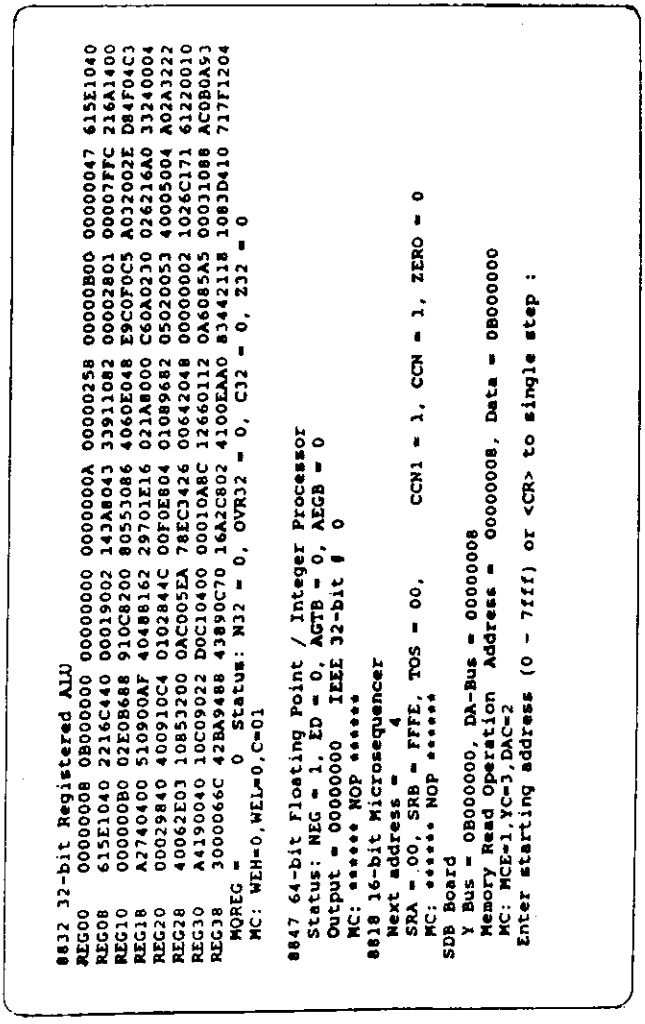

Figura 2. Pantalla del SDB

Descripción del diseño

Nuestro diseño (fig. 3) consiste de 13 registros de 16 dígitos binarios. Hay tres registros disponibles para el programador: dos de uso general (RB y RC) y el acumulador (ACC). Los 10 registros restantes que se describen a continuación son para propósitos de la microprogramación:

- "Program counter" (PC)

  El PC contiene la localización de memoria de la próxima instrucción a ejecutarse.

- "Memory address register" (MAR)

  El MAR contiene la localización de memoria de la próxima lectura o escritura.

- "Instruction register" (IR)

  El IR contiene la instrucción que se está ejecutando.

- "Memory data register" (MDR)

  El MAR contiene el operando que se lee de la memoria principal o el que se va a escribir en la memoria. Este operando es el segundo operando si se usa modo de dirección directo.

- "Data register" (DR)

  El DR contiene el segundo operando a usarse en una operación si se usa la dirección de memoria inmediata o de registro.

- "Stack pointer" (S)

El S almacena la dirección de memoria donde ocurre un brinco a una subrutina. Luego de ejecutar la operación obtiene el valor que contiene la localización de memoria que indica el "S" para continuar con la secuencia original del programa.

- Registro de banderas

  El registro de banderas indica el estado luego de realizar la operación de comparación (CMP). Sólo se usa el primer dígito binario e indica si el resultado fue cero. De ser así entonces contiene un número 1 en el primer dígito binario del registro.

- Registro decodificador OPCODE

  Este registro decodificador almacena el código operacional de la instrucción para luego determinar qué operación se va a realizar.

- Registro decodificador de modo de dirección

  Este registro decodificador almacena la información de la instrucción que indica el modo de dirección.

- Registro decodificador de registro

  Este registro decodificador almacena la información que determina cuál registro se va a usar.

El microprocesador que diseñamos usa dirección sencilla ("single addressing"). Por lo tanto, el acumulador siempre contiene el valor del primer operador. Luego el resultado de una operación se almacena en el acumulador y la información que había en el acumulador antes de la operación se destruye. Los registros de uso general se utilizan para almacenar datos temporalmente.

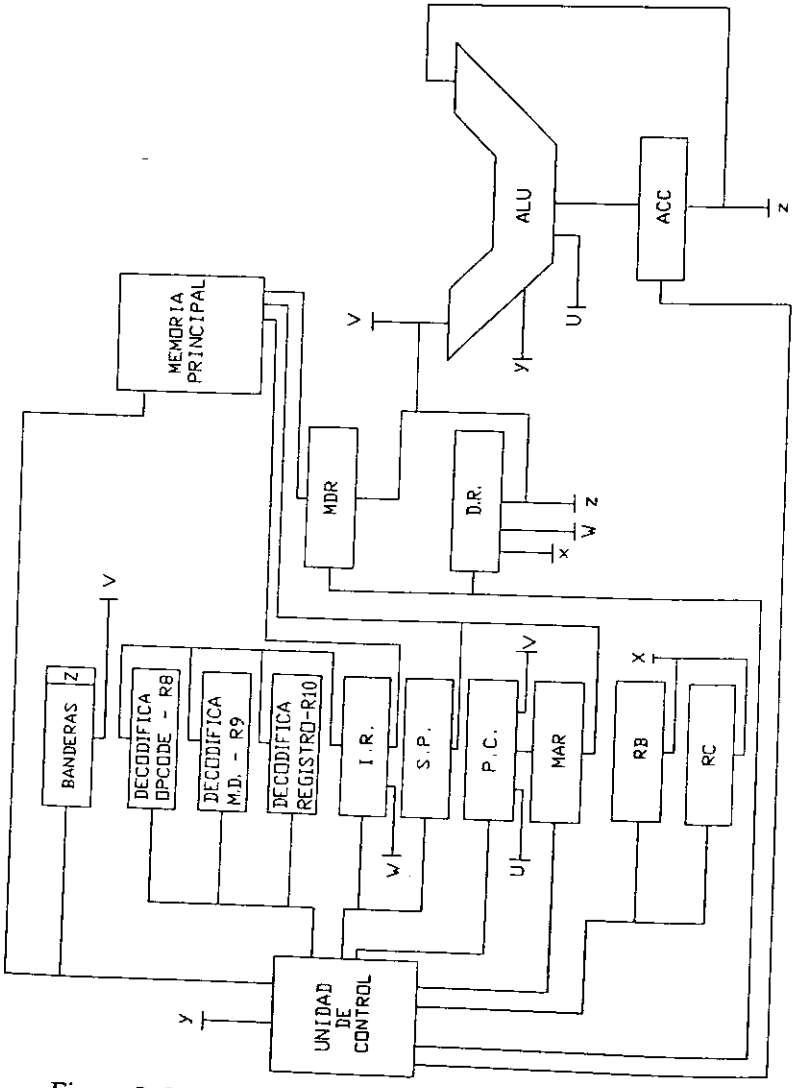

Figura 3. Diagrama del microprocesador diseñado

Conjunto de instrucciones

El conjunto de instrucciones del microprocesador consta de 24 instrucciones y cuatro modos de dirección. El formato para todas las instrucciones es de 16 dígitos binarios. Las instrucciones se describen en la tabla 2. Los modos de dirección son el modo inmediato, modo directo, modo de registro y el modo implícito.

Modo inmediato

En el modo inmediato las instrucciones contienen el valor del segundo operador que se utiliza para ejecutar la instrucción. El primer operador siempre reside en el acumulador. Las instrucciones que se pueden utilizar con este modo son: LOD, ADD, SUB, AND, OR, XOR, JP, BNZ, CMP, JPS. Los primeros dos dígitos binarios indican el modo de dirección (fig. 4), los próximos cinco dígitos indican el "OPCODE", los próximos dos indican el registro a usarse para la instrucción (para el modo inmediato no se usan registros) y los últimos siete indican el segundo operando.

Figura 4 - Formato de instrucciones para modo inmediato

Al usar estas instrucciones se debe indicar el tipo de instrucción, el modo de dirección, que no se usan registros con las siglas "NR" ("no register") y, por último, el operando. Por ejemplo, la instrucción LOD del modo inmediato se debe escribir de la siguiente manera:

LOD IM,NR,B#0000001

Esta instrucción almacena el número 1 en el acumulador. Los valores de los operandos deben ser números binarios que ocupen siete dígitos binarios.

Tabla 2 - Descripción del conjunto de instrucciones

Opcode	Operación	Descripción
LOD	Acc := [X]	Carga el acumulador
STO	X := [Acc]	Almacena el valor del acumulador
ADD	Acc := Acc + [X]	Súmale al acumulador
SUBS	Acc := Acc - [X]	Réstale al acumulador
MUL	Acc := (Acc)*([X])	Multiplícale al acumulador
DIV	Acc := Acc/[X]	Divídele al acumulador
AND	Acc := Acc AND [X]	Operación lógica AND con acumulador
OR	Acc := Acc OR [X]	Operación lógica OR con acumulador
XOR	Acc := Acc XOR [X]	Operación lógica XOR con acumulador
JP	PC := X	Brinca la secuencia del programa sin condición
BNZ	PC := X	Brinca la secuencia del programa con condición
JPS	PC := X	Brinca a una subrutina
CMP	{Acc := Acc - [X]} = 0	Compara el valor del acumulador
RT	PC := S	Continúa secuencia luego de realizar subrutina
NEG	Acc := - Acc	Valor negativo del acumulador
CMT	Acc := CMT [Acc]	Complementa el valor del acumulador
RTL	Acc := RTL [Acc]	Rota el valor del acumulador a la izquierda
RTR	Acc := RTR [Acc]	Rota el valor del acumulador a la derecha
SL	Acc := SL [Acc]	Desplaza el valor del acumulador a la izquierda
SR	Acc := SR [Acc]	Desplaza el valor el acumulador a la derecha
NOP	N/A	No se realiza operación
STP	PC := PC	Detiene el programa
DADD	Acc := Acc + [X]	Súmale números decimales al acumulador
DSUB	Acc := Acc - [X]	Réstale números decimales al acumulador

Para la instrucción BNZ, que significa "ramificar si no es cero", se utiliza primero la instrucción CMP ("compare"), que le resta al acumulador el valor del operando. Luego la operación BNZ verifica si el resultado de la resta es cero y si no es cero brinca a la localización de memoria que indica el operando. Cuando el resultado de la resta es cero, el programa continúa la secuencia normal.

Modo directo

En el modo directo las instrucciones contienen la dirección de memoria donde se encuentra el segundo operando. Estas instrucciones son: LOD, STO, ADD, SUB, MUL, DIV, AND, OR, XOR, DADD y DSUB. Los primeros dos dígitos binarios indican el modo de dirección, los próximos cinco dígitos indican el "OPCODE", los próximos dos indican cuáles registros se usarán para la instrucción (para el modo directo no se usan registros) y los últimos siete indican la dirección de memoria donde se encuentra el segundo operador (fig. 5).

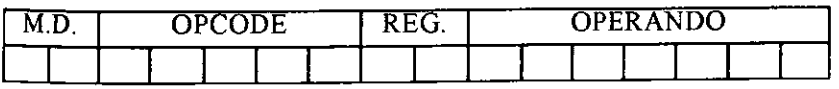

Figura 5 - Formato de instrucción para modo directo

Un ejemplo de una instrucción de modo directo es la siguiente:

```
LOD     DI,NR,B#0001111
```

Esta instrucción busca la localización de memoria 0F hexadecimal, obtiene el valor del operador que tiene almacenado y luego lo guarda en el acumulador.

Las instrucciones MUL, DIV, DADD, DSUB sólo se usan con números reales. Los números reales son el equivalente de notación científica en números binarios de computadora. En el formato de IEEE ("Institute of Electrical and Electronics Engineers") los números de punto flotante tienen

23 dígitos binarios de número fraccionario (f), ocho dígitos binarios de exponente (e) y un dígito binario que indica el signo (s). Debido a que el CPU diseñado es de 16 dígitos binarios, el formato tiene siete dígitos binarios fraccionarios solamente (fig. 6). El número 1.00 se representa como 3F00 hexadecimal en el formato de punto flotante.

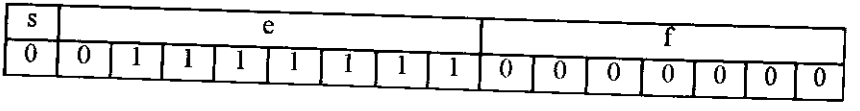

Figura 6 - Representación de IEEE de punto flotante

Modo de registros

En el modo de registros las instrucciones contienen el registro donde se encuentra el segundo operador. Estas instrucciones son LOD, STO, ADD, SUB, MUL, DIV, AND, OR, XOR, DADD y DSUB. Los primeros dos dígitos binarios indican el modo de dirección, los próximos cinco dígitos indican el "OPCODE", los siguientes dos indican el registros del cual se obtiene el operador y los últimos siete no se usan en este modo (fig. 7).

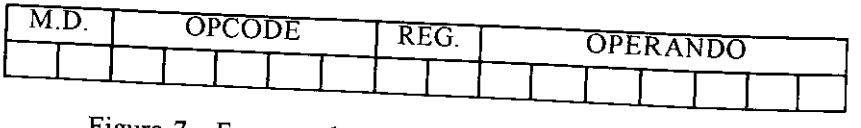

Figura 7 - Formato de instrucción para modo de registro

Por ejemplo, una instrucción de modo directo se escribe de la siguiente manera:

```
LOD     RE,RB
```

Esta instrucción obtiene el valor del operando del registro B (RB) y lo almacena en el acumulador. Al igual que en el modo de dirección directo, sólo se pueden usar números de punto flotante para las instrucciones MUL, DIV, DADD y DSUB.

Modo implícito

En el modo implícito las instrucciones no requieren un segundo operando para realizar la operación. Las instrucciones de éste modo ocasionan un cambio en el acumulador, por lo tanto, el único operando que necesitan se encuentra en el acumulador. Estas instrucciones son RT, NEG, CMT, RTL, RTR, SL, SR, NOP y STP. Los primeros dos dígitos binarios indican el modo de dirección, los próximos cinco dígitos indican el OPCODE, los últimos nueve dígitos tienen el valor de cero (fig. 8).

M.D.		OPCODE														
							0	0	0	0	0	0	0	0	0	0

Figura 8 - Formato de instrucción para modo implícito

Una instrucción de modo implícito se escribe de la siguiente manera:

NEG IP.

Esta instrucción le cambia el signo al valor del acumulador.

Ejecución de una instrucción

El ciclo de leer-decodificar-ejecutar comienza al obtener la dirección de la instrucción que se va a a leer, la cual está en la PC, y almacenarla en la dirección de memoria en el MAR. Luego se lee la instrucción de su localización de memoria y se almacena en el IR. Entonces la unidad de control decodifica la instrucción que se encuentra en el IR, determina el tipo de instrucción y la ejecuta. Para ir a la próxima instrucción le suma cuatro al PC. Los siguientes pasos describen el ciclo de leer - decodificar - ejecutar de una instrucción "ADD" del modo inmediato:

Lectura:

MAR := PC

IR := Valor de la instrucción que se encuentra en la memoria

Decodifica:

Decodifica la instrucción y determina que la operación es un "ADD" de modo inmediato

Ejecuta:

MAR := OPERADOR
ACC := ACC + MAR
PC := PC + 4

Microprogramación y fase de prueba

La microprogramación del CPU consiste de dos partes: el microcódigo y la tabla de definiciones para ensamblador. El microcódigo es la parte de la microprogramación que define las funciones del microprocesador. La tabla de definiciones para el ensamblador traduce el programa de lenguaje de ensamblaje a lenguaje de microprogramación. Para las pruebas escribimos dos programas en lenguaje ensamblaje. El primer programa prueba todas las instrucciones del microprocesador y el segundo resuelve la multiplicación de una matriz 2x2 con una matriz 2x3.

Conclusiones

Nuestro propósito principal al realizar este trabajo de investigación fue diseñar un microprocesador. Durante el proceso aprendimos mucho más sobre los microprocesadores que ejecutan procesos paso a paso. El modelo diseñado es sencillo y tiene las características de un microprocesador común. El proyecto fue un éxito ya que el microprocesador trabaja según se especificó en el diseño.

El emulador SDB 8800 fue una herramienta muy útil y necesaria en nuestro diseño porque fue el medio que usamos para corroborar físicamente cómo trabaja el microprocesador. Además, si se comete algún error en la microprogramación, se observa el efecto que causa y las formas de corregirlo. Esta tarjeta microprogramable es muy útil en el estudio de los microprocesadores.

Nuestro diseño se puede alterar de varias maneras. Por ejemplo, se le pueden añadir instrucciones para aumentar la capacidad de ejecución del microprocesador y reestructurar la microprogramación para que cada operación se realice más rápido o cambiar el formato de las instrucciones. También se pueden comprar los componentes (por ejemplo: registros, ALU, ROM) y construir el microprocesador.