Capítulo 4

Procesador

...

...

Resumen: En este capitulo se presenta el diseño del procesador implementado: su arquitectura,

4.1. Introducción

Se ha diseñado e implementado un procesador con arquitectura RISC basado en la arquitectura de los procesadores DLX estudiados durante el grado en ingeniería de computadores [10]. Se trata de un procesador con un ancho de palabra de 32 bits y una segmentación en 5 etapas.

La implementación ha sido adaptada para poder ejecutar instrucciones del repertorio ARM. En concreto, se permite ejecutar un subconjunto del juego de instrucciones THUMB-2. Este juego de instrucciones es utilizado principalmente por los procesadores de la gama ARM CORTEX M.

Para el desarrollo de este proyecto se ha utilizado la tecnología de las FPGAs, en concreto la placa «Nexys 4» que hace uso de la FPGA «Artix 7» de Xilinx.

Para la implementación se ha utilizado el lenguaje de diseño hardware VHDL junto al software «ISE Design Suite 14.4» de Xilinx para la implementación y el software «ModelSim PE Student Edition» de Altera para las simulaciones. Con esto se ha conseguido probar el diseño y la implementación de forma rápida y realizar las correcciones necesarias.

4.2. Arquitectura del procesador

El microprocesador es una implementación modificada de la arquitectura DLX para permitir ejecutar instrucciones del repertorio ARM. A continuación se describen sus características completas.

4.2.1. Estructura

El procesador se compone de los siguientes elementos (figura 4.1):

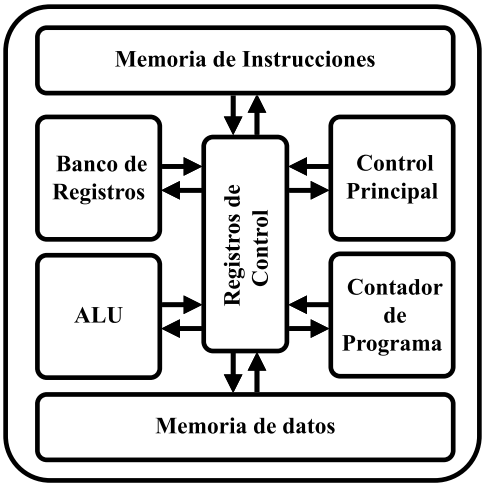

Figura 4.1: Estructura de procesador diseñado

- **Banco de registros:** Dispone de 16 registros (R0, R1, ..., R15) de propósito general con un tamaño de 32 bits. Estos registros se pueden utilizar tanto para guardar datos leídos de memoria como enviar los valores a memoria. Se puede trabajar con los valores que tengan almacenados ejecutando operaciones sobre ellos. El registro R15 es accesible de forma limitada puesto que el identificador de este registro se utiliza para diferenciar unas instrucciones de otras.
- **Contador de programa (PC):** Este registro especial almacena la dirección de memoria de la instrucción que debe ejecutarse a continuación. Se incrementa automáticamente en 4 cada ciclo y solo se puede alterar este mecanismo por medio de instrucciones de control.
- **Memoria de instrucciones:** Esta memoria almacena el código de programa, es accedido por el procesador para analizar y ejecutar las instrucciones.
- **Memoria de datos:** Esta memoria conserva los valores de los datos que son accesibles por el procesador mediante instrucciones de carga y

almacenamiento.

- **Control principal:** Analiza las instrucciones para extraer la información necesaria para ejecutar las mismas adecuadamente. Esta información se propaga medante señales de control al resto de componentes.

- **Unidad Aritmético-Lógica (ALU)**: Es el componente encargado de realizar las operaciones aritméticas o lógicas sobre los operandos.

- **Registros de control:** Almacena las señales de control, y las señales internas entre las diferentes etapas de la segmentación.

4.2.2. Repertorio de instrucciones

El procesador implementado es capaz de ejecutar 3 tipos de instrucciones:

- Accesos a memoria

- Procesamiento de datos

  a). Operaciones con dos registros

  b). Operaciones con un registro y un inmediato

- Operaciones de control

A continuación se explican brevemente los diferentes tipos de instrucciones. Más adelante se expondrán las instrucciones con más detalle, explicando los campos de cada una.

Accesos a memoria

Las instrucciones de acceso a memoria son necesarias cuando se requiere cargar (load) un dato desde la memoria al banco de registros, o almacenar (store) el valor de un registro en la memoria.

Aunque es posible acceder a las direcciones de memoria direccionadas por media palabra. En esta implementación se está obligado a cargar valores de tamaño 4 bytes (tamaño de palabra), siendo por tanto recomendable utilizar direcciones de memoria que sean múltiplos de 4.

Para el cálculo de la dirección efectiva de carga o almacenamiento se ha implementado un único modo de direccionamiento. Registro base Rn + imm12", es decir, la dirección base se obtiene del registro Rn, y se suma un inmediato de 12 bits extraído de la instrucción.

Procesamiento de datos

Las instrucciones de procesamiento realizan cálculos aritméticos y lógicos. Se aplican sobre dos operandos y el resultado (si existe) se almacena en un registro.

Dependiendo de la instrucción los operandos pueden ser:

- **Operaciones con dos registros:**

  Los datos con los que se trabaja se extraen de dos registros.

  Al utilizar el registro R15 se deben tener en cuenta ciertas restricciones. Este registro se utiliza para diferenciar ciertas operaciones de otras. Por ejemplo, si el código de operación es "0010", el registro origen Rn es R15 ("1111") entonces la operación ejecutada será la operación "MOVE". Si el registro Rn es cualquier otro, se ejecutará una "Ó lógica"(operación or).

- **Operaciones con un registro y un inmediato:**

  El conjunto de operaciones con inmediato se limita a cuatro. Se permite mover un inmediato a la mitad más significativa, o a la menos significativa, de un registro. Y se permite sumar o restar un inmediato al valor de un registro.

Operaciones de control

Las operaciones de control intervienen en la ejecución normal del programa. Se utilizan para modificar el valor del registro del contador de programa, esta operación se conoce como una instrucción de salto.

Existen dos tipos de instrucciones de salto. El primero es el salto incondicional y permite sumar un entero al valor del contador de programa y almacenar el resultado en el mismo.

La segunda operación de control es el salto condicional. Previamente a un salto condicional se debe ejecutar una operación de comparación para actualizar los flags de comparación de la ALU. Los flags se comparan a la condición de salto y en caso de coincidir, se efectúa el salto. Si no se ejecuta la comparación, el estado de los flags es desconocido y el procesador se comportará de manera no controlada.

4.2.3. Segmentación

Se ha segmentado el microprocesador en 5 etapas, cada una de ellas realiza una parte fundamental en la ejecución de las instrucciones. Las etapas en las que se divide el procesador son:

1. **Búsqueda de instrucción (IF):**

La primera etapa es la encargada de cargar las instrucciones de memoria y transmitirlas a la siguiente, simultáneamente se calcula la dirección de la siguiente instrucción.

2. **Decodificación de instrucción(ID):**

   En la etapa de decodificación se analiza la instrucción y se obtienen los datos necesarios para realizar las operaciones correctamente.

3. **Ejecución (EX)**

   En la etapa de ejecución se realizan los cálculos aritméticos o lógicos sobre los datos obtenidos del banco de registro y del circuito de extensión de signo.

4. **Acceso a Memoria (MEM)**

   En la etapa de memoria se realizan intercambios de datos con la memoria principal.

5. **Escritura en registros (WB)**

   En la etapa final del procesador se escriben los resultados calculados por la ALU, o los datos cargados de memoria en el banco de registros.

4.2.4. Memoria

El diseño del procesador hereda el sistema de memoria de la arquitectura Harvard, es decir, tiene acceso a dos memorias diferentes:

- **Memoria de instrucciones:**

  Memoria ROM donde se almacenan todas las instrucciones del programa a ejecutar.

- **Memoria de datos:**

  Memoria RAM accesible en modo lectura y en modo escritura para almacenar los datos con los que trabaja el programa.

4.3. Implementación

La implementación del proyecto se ha realizado utilizando la herramienta «ISE Design Suite» de Xilinx y el lenguaje de diseño hardware «VHDL». En esta sección se describen los componentes principales del procesador (figura 4.1).

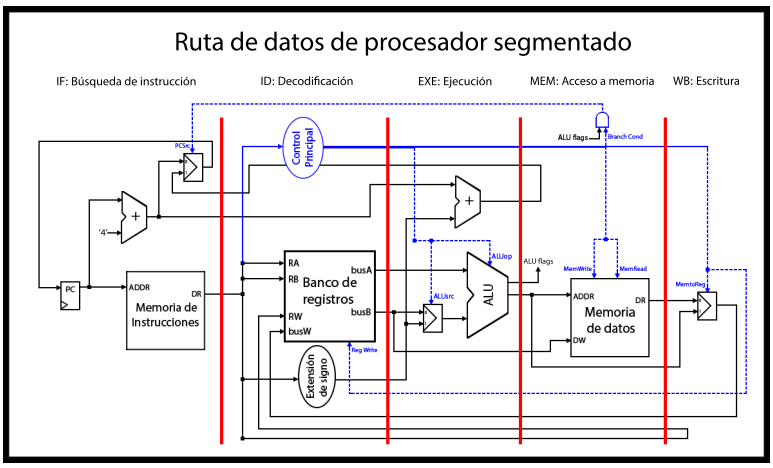

Figura 4.2: Diseño completo del procesador segmentado.

4.3.1. Banco de registros

El banco de registros es el componente de memoria con el que opera principalmente el procesador. De este obtiene los datos con los que realiza la mayoría de las operaciones.

Internamente se compone de 16 registros que almacenan valores de 32 bits. El acceso a estos se codifica en 4 bits de modo que el registro accedido con el valor «1010» es el registro «R10».

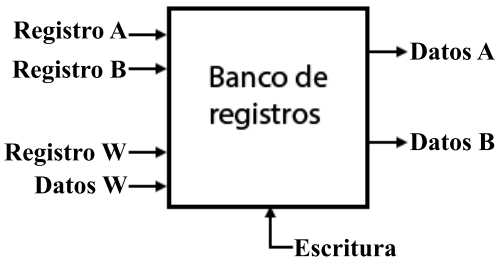

Figura 4.3: Banco de registros.

Sus señales externas son (figura 4.3):

- **Señales de entrada**
  - **Registro A:** Registro origen del primer operando de la instrucción.

- **Registro B:** Registro origen del segundo operando de la instrucción.
- **Registro W:** Registro donde se almacenará el resultado de la instrucción.
- **Datos W:** El valor que se almacenará en el registro W.
- **Escritura:** Habilita la escritura en el registro W.

▪ **Señales de salida**

- **Datos A:** Valor del registro A, primer operando de la instrucción.
- **Datos B:** Valor del registro B, segundo operando de la instrucción.

4.3.2. Contador de programa

El contador de programa es un registro común de 32 bits encargado de almacenar la dirección de memoria que indica donde se encuentra la siguiente instrucción del programa. Su valor se actualiza en cada ciclo de reloj.

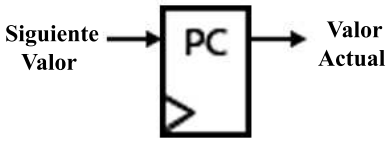

Figura 4.4: Contador de programa.

Sus señales externas son (figura 4.4):

▪ **Señales de entrada**

- **Siguiente Valor:** El valor de la siguiente instrucción. Puede ser calculada de forma secuencial, o por medio de un salto efectivo.

▪ **Señales de salida**

- **Valor actual:** Dirección origen de la instrucción que debe ejecutarse.

4.3.3. Unidad Aritmético-Lógica (ALU)

La unidad aritmético-lógica aplica ciertas operaciones sobre los datos extraídos previamente del banco de registros y de la instrucción. Así mismo, y

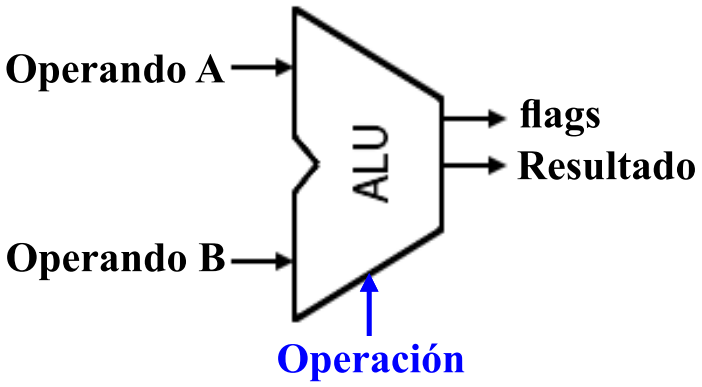

Figura 4.5: Unidad aritmético-lógica.

dependiendo del resultado, puede modificar el valor de los flags de comparación que se modifican solo si la instrucción así lo requiere.

Las señales externas son (figura 4.5):

- **Señales de entrada**
  - **Operando A:** Primer operando de la operación.
  - **Operando B:** Segundo operando de la operación.
  - **Operación:** Operación que debe aplicarse sobre los operadores.
- **Señales de salida**
  - **flags:** Indican si el resultado de la operación cumple ciertas condiciones[1].
  - **Resultado:** Valor de aplicar la operación a los operandos.

4.3.4. Control principal

El control principal del microprocesador es el encargado de analizar la instrucción y establecer las señales de control que permitirán a los módulos realizar la función adecuada.

Éste es un módulo combinacional, por lo tanto analiza y establece los valores de las señales de control dentro de un único ciclo de reloj.

La instrucción es la una única entrada para este módulo, sin embargo, las señales de control son varias y se transmiten de etapa a etapa de la segmentación hasta que son consumidas. A continuación se explican por orden en el que son utilizadas(figura 4.2):

[1] Si el resultado de la operación es igual a cero se activa el flag correspondiente.

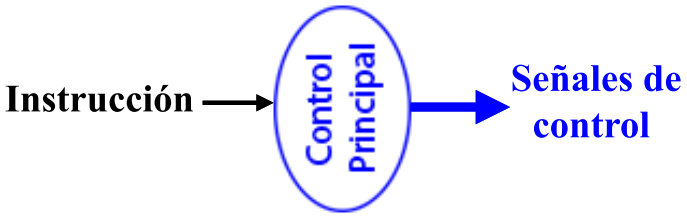

Figura 4.6: Control principal.

1. **Ejecución (EX)**
   - **ALUsrc:** Establece el origen del segundo operando de la instrucción, este puede ser un registro o la propia instrucción.
   - **ALUop:** Asigna a la ALU la operación que debe aplicar a los operandos.
2. **Acceso a Memoria (MEM)**
   - **Branch Cond:** Junto a los flags de la ALU, establece la señal **PCSrc** encargada de efectuar un salto en el código del programa.
   - **MemWrite:** Indica al módulo «memoria de datos» si debe acceder a memoria en modo escritura[2].
   - **MemRead:** Indica al módulo «memoria de datos» si debe acceder a memoria en modo lectura[2].
3. **Escritura en registros (WB)**
   - **MemtoReg:** Establece el origen de los datos que deben almacenarse en el banco de registros, puede ser la ALU o la memoria de datos.
   - **RegWrite:** Indica al banco de registros si el resultado de la instrucción debe almacenarse en un registro.

4.3.5. Registros de control

Los registros de control son todos aquellos que almacenan la información entre las etapas (lineas rojas en figura 4.2).

Esta colección de registros son fundamentales para que exista la segmentación. Separan las etapas de forma que los cambios en una de ellas no se propaguen y no interfieran con las demás etapas.

[2] El modo lectura y el modo escritura en memoria son incompatibles, solo debe activarse una de estas señales.

Para que esto no suceda se inserta una serie de registros entre, por ejemplo, los componentes de las etapas de decodificación y ejecución. Éstos registros se actualizan al final de cada ciclo de reloj, conservando los datos de forma estable para que se puedan utilizar correctamente en la siguiente etapa. Así se permite que ambas etapas trabajen de forma independientemente.

Figura 4.7: Registros de control

4.3.6. Memoria de instrucciones

La memoria de instrucciones es una memoria de solo lectura (ROM) Este tipo de memorias, como su nombre indica, solo permite la lectura de sus datos, y no permite que se modifiquen. Este módulo permite que se consulte el valor de la dirección en un único ciclo de reloj.

Las señales externas de esta memoria son (figura 4.9):

- **Señales de entrada**
  - **Dirección:** Dirección de la instrucción a la que se accede.
- **Señales de salida**
  - **Instrucción:** Instrucción leída de memoria.

4.3.7. Memoria de datos

La memoria de datos es el encargado de almacenar aquellos datos que no son inmediatamente necesarios para ejecutar las instrucciones. Es de un

Figura 4.8: Memoria de Instrucciones.

mayor tamaño que el banco de registros y, por lo general, son más lentos a la hora de transmitir la información.

Este módulo se ha implementado como un banco de registros de mayor tamaño que el módulo con el mismo nombre, pero de mayor tamaño. Su acceso se completa en un mismo ciclo de reloj.

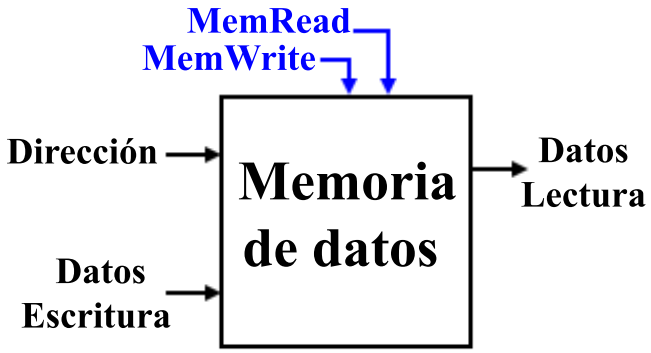

Figura 4.9: Memoria de datos.

Las señales externas de esta memoria son (figura 4.9):

- **Señales de entrada**
  - **Dirección:** Dirección de acceso a memoria, ya sea en modo lectura o escritura.
  - **Datos Escritura:** Datos que deben almacenarse en la memoria.

- **MemWrite:** Indica que los datos deben escribirse en memoria[3].
- **MemRead:** Indica que los datos deben leerse de memoria[3].

- **Señales de salida**
  - **Datos Lectura:** Datos que se han leído de la dirección indicada de memoria.

4.4. Formato de instrucciones

En esta sección se exponen el formato de las instrucciones que es capaz de ejecutar el microprocesador, con todos sus campos y el significado de estos.

4.4.1. Accesos a Memoria

Estas instrucciones permiten al microprocesador acceder a los datos almacenados en la memoria de datos así como almacenar datos en ella. Las instrucciones de carga o almacenamiento se identifican por los 7 bits más significativos de la instrucción, estos deben ser «1111100».

Se ha implementado un único tipo de instrucción de acceso a memoria. Éste, dependiendo del valor de sus campos, se utiliza para cargar de o almacenar datos en memoria.

Esta instrucción permite realizar transferencias de datos entre el banco de registros y la memoria de datos del microprocesador. Permite aplicar un desplazamiento de hasta 4KB al valor base del registro.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Formato general	1	1	1	1	1	0	0									
Rn + imm12								S	1	Size		L	Rn			

Tabla 4.1: Instrucciones de acceso a memoria (bits 31..16)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Formato general																
Rn + imm12	Rd				imm12											

Tabla 4.2: Instrucciones de acceso a memoria (bits 15..0)

Los campos de la instrucción «Rn + imm12» son los siguientes:

- **Extensión de signo (S):** Indica si se debe extender el signo del valor inmediato (S=1).

[3] El modo lectura y el modo escritura en memoria son incompatibles, solo debe estar activa una de estas señales.

- **Tamaño (Size):** Indica el tamaño del valor que debe cargar de o almacenar en memoria. No se utiliza, se carga un dato de tamaño igual al tamaño de palabra.
- **Cargar/Almacenar (L):** Este bit indica si la operación debe leer un dato de memoria (L=1) o escribirlo (L=0).
- **Dirección (Rn):** Indica el registro con la dirección base de acceso a memoria.
- **Dato (Rt):** Indica el registro donde se debe almacenar el dato en caso de carga, o el registro donde debe extraerse el dato en caso de almacenamiento.
- **Desplazamiento (imm12):** El desplazamiento que debe aplicarse a la dirección base para obtener la dirección efectiva.

4.4.2. Procesamiento de datos

Las operaciones se separan según el tipo de operandos que se apliquen. El operando A siempre es obtenido de un registro. Mientras que el operando B puede ser el valor de un segundo registro o puede formar parte de la instrucción.

Operaciones con dos registros

Las operaciones que hacen uso de dos registros son aritméticas (sumar, restar y mover), lógicas (and, or y or exclusiva) y de comparación que activan diferentes flags (Negativo, Cero). En el caso de una comparación no se modifican los registros. Las instrucciones de operación con dos registros se identifican por los 7 bits más significativos. Éstos deben ser «1110101».

Se ha implementado un único tipo de instrucción de procesamiento con dos registros. Dependiendo del valor de sus campos se aplicará una operación u otra sobre los operandos.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Formato general	1	1	1	0	1	0	1									
Data Processing								OP				S	Rn			

Tabla 4.3: Instrucciones de procesamiento de datos con dos registros (bits 31..16)

Los campos de la instrucción «Data Processing» son los siguientes:

- **Código de operación (OP):** Indica la operación que debe aplicarse en la fase de ejecución sobre los operandos.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Formato general																
Data Processing	SBZ	imm3			Rd				imm2		type		Rm			

Tabla 4.4: Instrucciones de procesamiento de datos con dos registros (bits 15..0)

- **Activar flags (S):** Indica si deben activarse los flags de salto al ejecutar la operación.
- **Registro origen A (Rn):** Indica el registro origen del primer operando.
- **Should be Zero (SBZ):** Este campo debe tener un valor de 0.
- **Inmediato (imm3:imm2):** Indica el desplazamiento que debe aplicarse al segundo operando. No se utiliza.
- **Registro destino (Rd):** Indica el registro destino donde se almacenará el resultado de la operación.
- **Tipo de desplazamiento (type):** Indica el tipo de desplazamiento aplicado. No se utiliza.
- **Registro origen B (Rm):** Indica el registro origen del segundo operando.

Las instrucciones implementadas junto con sus respectivos códigos de operaciones se muestran en la tabla 4.5.

Operación	Código	Restricciones
ADD	1 0 0 0	
AND	0 0 0 0	
CMP	1 1 0 1	(Rd==«1111»,S==1)
EOR	0 1 0 0	
MOV	0 0 1 0	(Rn==«1111»)
ORR	0 0 1 0	
SUB	1 1 0 1	

Tabla 4.5: Operaciones con dos registros

Operaciones con un registro y un inmediato

Las operaciones que hacen uso de un registro y un inmediato únicamente pueden ser aritméticas (sumar, restar y mover). Estas instrucciones se dividen en dos tipos según el tamaño del inmediato utilizado. Para identificar

este tipo de instrucciones se utilizan los 5 bits más significativos, que deben ser «11110» y el bit 15 que debe valer «0».

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Formato general	1	1	1	1	0											
Add, Subtract, plain 12-bit immediate						i	1	0	OP	0	OP2		Rn			
Move, plain 16-bit immediate						i	1	0	OP	1	OP2		imm4			

Tabla 4.6: Instrucciones de procesamiento de datos con un registro y un inmediato (bits 31..16)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Formato general	0															
Add, Subtract, plain 12-bit immediate		imm3			Rd				imm8							
Move, plain 16-bit immediate		imm3			Rd				imm8							

Tabla 4.7: Instrucciones de procesamiento de datos con un registro y un inmediato (bits 15..0)

Los campos de las instrucciones implementadas son:

- **«Add, Subtract, plain 12-bit immediate»**
  - **Código de operación (OP:OP2):** Indica la operación que debe aplicarse en la fase de ejecución sobre los operandos.
  - **Registro origen (Rn):** Indica el registro origen del primer operando.
  - **Inmediato (i:imm3:imm8):** Contiene el inmediato que se utiliza como segundo operando.
  - **Registro destino (Rd):** Indica el registro destino donde se almacenará el resultado de la operación.
- **«Move, plain 16-bit immediate»**
  - **Código de operación (OP:OP2):** Indica la operación que debe aplicarse en la fase de ejecución sobre los operandos.
  - **Inmediato (imm4:i:imm3:imm8):** Contiene el inmediato que se utiliza como segundo operando. Utilizado en las operaciones mover.
  - **Registro destino (Rd):** Indica el registro destino donde se almacenará el resultado de la operación.

Las instrucciones implementadas junto con sus respectivos códigos de operaciones se muestran en la tabla 4.8.

Operación	Código
ADD	0 0 0
SUB	1 1 0
MOVT	1 0 0
MOV	0 0 0

Tabla 4.8: Operaciones con un registro y un inmediato

4.4.3. Operaciones de control

También conocidas como instrucciones de salto, estas instrucciones son aquellas capaces de alterar el contador de programa. Para identificar este tipo de instrucciones se utilizan los 5 bits más significativos, que deben ser «11110» y el bit 15 que debe tener un valor de «1».

Se han implementado dos tipos de instrucciones de salto: salto incondicional, y salto incondicional. El salto incondicional se realiza siempre que esté presente la instrucción. La operación de salto condicional sólo se efectúa cuando coinciden las condiciones de la instrucción con los flags previamente calculados de la ALU.

En el caso del salto incondicional permite realizar un salto de 16MB por el código. El salto condicional, debido a necesitar un campo que indique la condición, puede realizar un salto de 1MB.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Formato general	1	1	1	1	0											
Branch						S	offset[21:12]									
Conditional Branch						S	cond				offset[17:12]					

Tabla 4.9: Instrucciones de control (bits 31..16)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Formato general	1															
Branch		0	I1	1	I2	offset[11:1]										
Conditional Branch		0	J1	0	J2	offset[11:1]										

Tabla 4.10: Instrucciones de control (bits 15..0)

Los campos de las instrucciones de salto implementadas son:

- **Salto (Branch)**
  - **Extensión de signo (S):** Indica si se debe extender el signo del desplazamiento (S=1).
  - **Desplazamiento (offset):** Contiene el inmediato que se suma al registro PC para calcular la direcció efectiva del salto. Los campos

«I1» e «I2» son los bits 23 y 22 del desplazamiento respectivamente.

- **Salto condicional (Conditional Branch)**
  - **Extensión de signo (S):** Indica si se debe extender el signo del desplazamiento (S=1).
  - **Condición de salto (cond):** Indica la condición necesaria para que el salto deba realizarse.
  - **Desplazamiento (offset):** Contiene el inmediato que se suma al registro PC para calcular la dirección efectiva del salto. Los campos «J1» e «J2» son los bits 19 y 18 del desplazamiento respectivamente.

Bibliografía

Y así, del mucho leer y del poco dormir,
se le secó el celebro de manera que vino
a perder el juicio.

Miguel de Cervantes Saavedra

[1] R. Brinkgreve, W. Swolfs, and E. Engin. *ARM Architecture Reference Manual Thumb-2 Supplement.* 2011.

[2] S. Brown and J. Rose. Architecture of FPGAs and CPLDs: A tutorial. *IEEE Design and Test of Computers*, 13(2):42–57, 1996.

[3] C. T. Bustillos. Simulador arm en el ámbito docente. 2012.

[4] I. N. de Estadistica. Penetración de ordenador en hogares. 2014.

[5] S. Flash. Nexys4 FPGA Board Reference Manual Ethernet connector. pages 1–29, 2013.

[6] J. Gaisler. A portable and fault-tolerant microprocessor based on the SPARC V8 architecture. *Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 409–415, 2002.

[7] J. C. González Salas. *Filtro adaptativo tolerante a fallos.* PhD thesis, 2014.

[8] S. Habinc. Functional Triple Modular Redundancy (FTMR). *Design and Assessment Report, Gaisler Research*, pages 1–56, 2002.

[9] J. L. Hennessy and D. A. Patterson. *Arquitectura de Computadores: Un enfoque cuantitativo.* Mcgraw Hill Editorial, 1993.

[10] J. L. Hennessy and D. a. Patterson. *Computer Architecture, Fourth Edition: A Quantitative Approach.* Number 0. 2006.

[11] A. C. Hu and S. Zain. NSEU Mitigation in Avionics Applications. 1073:1–12, 2010.

[12] O. Ieee-std. LEON3 7-Stage Integer Pipeline. (March), 2010.

[13] A. O. Investigation. ATSB TRANSPORT SAFETY REPORT Aviation Occurrence Investigation AO-2008-070 Final. (October), 2008.

[14] Jedec. Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray Induced Soft Error in Semiconductor Devices: JESD89A. *JEDEC Sold State Technology Association*, pages 1–85, 2006.

[15] A. Kadav, M. J. Renzelmann, and M. M. Swift. Fine-grained fault tolerance using device checkpoints. *Proceedings of the eighteenth international conference on Architectural support for programming languages and operating systems - ASPLOS '13*, page 473, 2013.

[16] H. Kirrmann. Fault Tolerant Computing in Industrial Automation. *Lecture notes ABB Corporate ResearchETH*, 2005.

[17] I. Kuon, R. Tessier, and J. Rose. FPGA Architecture: Survey and Challenges. *Foundations and TrendsÂ® in Electronic Design Automation*, 2(2):135–253, 2007.

[18] A. R. M. Limited. ARM7TDMI-S. (Rev 3), 2000.

[19] a. R. M. Limited. ARM Architecture Reference Manual. pages 1–1138, 2007.

[20] W. K. Melis. *Reconstruction of High-energy Neutrino-induced Particle Showers in KM3NeT .* PhD thesis, 2014.

[21] C. Mobile. Streaming 4K Ultra HD video at home and on the go. pages 0–1.

[22] J. Rose, A. E. Gamal, and A. Sangiovanni-Vincentelli. Architecture of Field-Programmable Gate Arrays.

[23] E. Rotenberg. AR-SMT: a microarchitectural approach to fault tolerance in microprocessors. *Digest of Papers. Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing (Cat. No.99CB36352)*, 1999.

[24] D. J. Sorin and S. Ozev. Fault Tolerant Microprocessors for Space Missions. *Memory*, pages 1–4.

[25] U. States. Reduce Cost and Board Space. 374:1–8, 2011.

[26] I. S. Summary, T. C. Field, M. Long, S. D. Transfer, U. Instruction, and I. S. Examples. ARM Instruction Set. pages 1–60.

[27] J. M. Torrecillas. RAID - Tolerancia a Fallos.

[28] C. Weaver and T. Austin. A fault tolerant approach to microprocessor design. *Proceedings of the International Conference on Dependable Systems and Networks*, (July):411–420, 2001.

[29] Xilinx. Xilinx Artix-7 Fpgas: a New Performance Standard for Power-Limited, Cost-Sensitive Markets.