

Capítulo 4

Procesador

...

...

RESUMEN: En este capítulo 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):

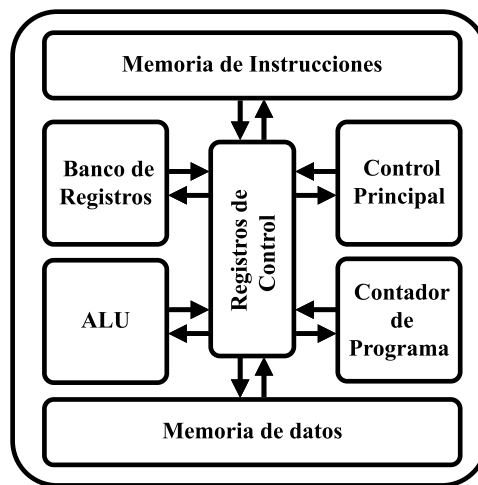


Figura 4.1: Estructura de procesador diseñado

- **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 mediante 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 juego de instrucciones elegido está compuesto por instrucciones de 32 bits con formato variable. El formato de las diferentes instrucciones se explicarán más adelante.

A diferencia de la arquitectura DLX, caracterizada por emplear instrucciones sencillas, se ha optado por utilizar un juego de instrucciones de mayor complejidad, por lo que se requiere una unidad de control compleja para decodificarlas. En la figura 4.2 se puede observar el formato equivalente para una misma instrucción de los repertorios DLX y ARM.

Figura 4.2: Comparación de instrucciones DLX y ARM

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

- Accesos a memoria
- Operaciones sobre registros
 - a). Operaciones con dos registros
 - b). Operaciones con un registro y un inmediato
- Operaciones de salto

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 $R_n + \text{imm12}$ ", es decir, la dirección base se obtiene del registro R_n , 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:

Figura 4.3: Formato para instrucciones aritmético-lógicas.

■ Operaciones con dos registros:

Los datos con los que se trabaja se extraen de dos registros codificados en 4 bits.

Al utilizar el registro R_{15} 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 R_n es R_{15} ("1111") entonces la operación ejecutada será la operación "MOVE". Si el registro R_n 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 de 16 bits a un registro, pudiendo elegir si los dos bytes se almacenarán en los 16 bits más significativos o en los 16 bits menos significativos. Además se permite sumar o restar un inmediato de 12 bits a 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.

Los procesadores ARM combinan instrucciones de 32 bits con instrucciones de 16 bits. Por ello, el inmediato es desplazado un bit hacia la izquierda. En nuestro caso nos debemos asegurar de codificar las instrucciones con un 0

en el bit menos significativo del inmediato. Con esto se evita acabar en una dirección equivocada, y leer dos mitades de dos instrucciones distintas.

Existen dos tipos de instrucciones de salto. El primero es el salto incondicional y permite sumar un entero de 24 bits al valor del contador de programa y almacenar el resultado en el mismo.

La segunda operación de control es el salto condicional. Para este tipo de salto se reduce el tamaño del inmediato a 20 bits. Esto es debido a que campo extra para la condición de salto, el tamaño de este es de 4 bits.

Previamente a un salto condicional se debe ejecutar una operación de comparación. Esta operación activa los flags de la unidad aritmético-lógica dependiendo del resultado de la comparación, y estos se mantienen hasta que se vuelva a ejecutar otra comparación. 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.

Figura 4.4: Formato para instrucciones de control.

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 transmitir las 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

La memoria de nuestro procesador se divide en dos componentes distintos:

- **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.

Bibliografía

*Y así, del mucho leer y del poco dormir,
se le secó el cerebro 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 Estadística. 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 Solid 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 Research ETH*, 2005.
- [17] I. Kuon, R. Tessier, and J. Rose. FPGA Architecture: Survey and Challenges. *Foundations and Trends® 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.