Capítulo 2

Introducción al procesador

«La idea detrás de los computadores digitales puede explicarse diciendo que estas máquinas están destinadas a llevar a cabo cualquier operación que pueda ser realizado por un equipo humano.»

Alan Turing

Resumen: En este capítulo se define con detalle lo que es un procesador y su importancia en el mundo de hoy en día. También se introduce la arquitectura ARM.

2.1. Procesador

El Diccionario de la Real Academia Española (DRAE) define el procesador como la «Unidad Central de Proceso (CPU), formada por uno o dos chips». Figura 2.1.

La CPU es el circuito integrado encargado de acceder a las instrucciones de los programas informáticos y ejecutarlas. Para poder ejecutar un programa, el procesador debe realizar las siguientes tareas:

1. Acceder a las instrucciones almacenadas en memoria.
2. Analizar las instrucciones y establecer las señales de control internas.
3. Ejecutar operaciones sobre datos.
4. Almacenar los resultados en memoria.

A continuación se definen los elementos fundamentales para constituir un procesador.

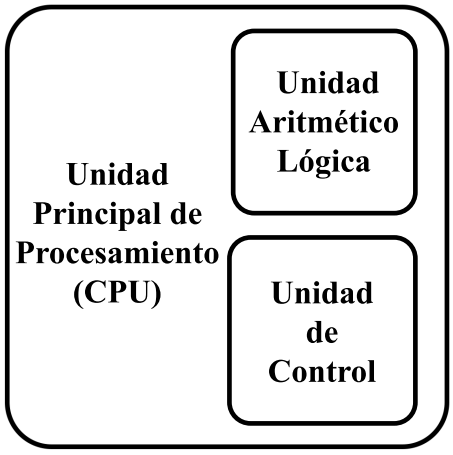

Figura 2.1: Procesador

2.1.1. Arquitectura

Un procesador está formado por una serie de módulos conectados entre sí, siendo la arquitectura del mismo la que define el diseño de los módulos que lo componen y de qué manera se conectan entre ellos.

La arquitectura del procesador diseñada por Von Neumann separa los componentes del procesador en módulos básicos. La CPU es el verdadero núcleo de los computadores, donde se realizan las funciones de computación y control. En la CPU se concentran todos los componentes, la memoria y los elementos de entrada/salida [8].

Según el juego de instrucciones que sea capaz de ejecutar un procesador, su arquitectura puede clasificarse como:

1. *Reduced instruction set computer (RISC)*. Utiliza un repertorio de instrucciones reducido, con instrucciones de tamaño fijo y poca variedad en su formato.

2. *Complex instruction set computer (CISC)*. Utiliza un repertorio de instrucciones muy amplio, permite realizar operaciones complejas entre las que se encuentran las de realizar cálculos entre los datos en memoria y los datos en registro.

2.1.2. Repertorio de instrucciones

El repertorio de instrucciones define todas las operaciones que el procesador es capaz de entender y ejecutar. Este juego de instrucciones incluye las operaciones aritmético-lógicas que pueden aplicarse a los datos, las operaciones de control sobre el flujo del programa, las instrucciones de lectura y

escritura en memoria, así como todas las instrucciones propias que se hayan diseñado para el procesador.

2.1.3. Memoria

Los procesadores tienen una serie de registros donde se almacenan temporalmente los valores con los que está trabajando. El conjunto de estos registros se conoce como «banco de registros». Los registros de propósito general son muy limitados, por lo que el procesador necesita disponer de apoyo externo donde alojar la información, para ello tiene acceso a una memoria externa.

El modo de acceso a la memoria externa divide las arquitecturas en dos tipos conocidas con los nombres de Von Neumann y Harvard. La arquitectura Von Neumann utiliza una única memoria para almacenar tanto los datos como las instrucciones. La arquitectura Harvard, sin embargo, separa la memoria de datos de la memoria de instrucciones. Figura 2.2.

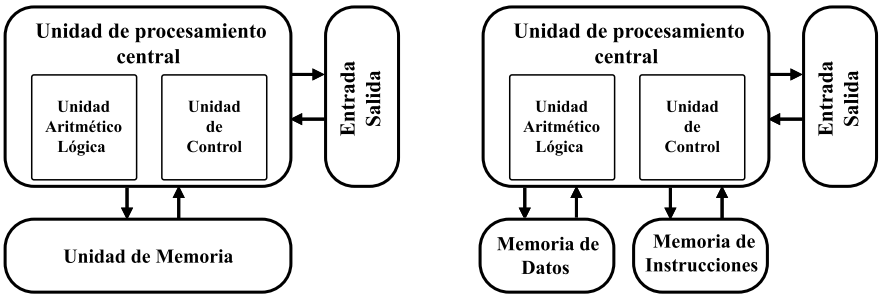

Figura 2.2: Arquitectura Von Neumann y Arquitectura Harvard

2.1.4. Segmentación

La segmentación consiste en dividir el procesador en etapas, de modo que en cada una de ellas se procesa una parte de una instrucción. Al dividir las instrucciones se consigue que cada etapa procese de forma independiente una parte de las mismas.

Las instrucciones van avanzando de etapa en etapa hasta que se terminen de procesar. De este modo se pueden tener en el procesador varias instrucciones ejecutándose de forma simultanea en distintas etapas, lo que resulta en un aumento significativo del rendimiento del procesador.

En la tabla 2.1 podemos ver cómo se procesan una serie de instrucciones en 5 etapas (IF, ID, EX, MEM, WB). Se observa cómo cada instrucción va ocupando una única etapa en cada ciclo de reloj, cómo cambian de una a otra al ser procesadas, permitiendo la ejecución de la siguiente instrucción.

	Ciclo de reloj								
Número de instrucción	1	2	3	4	5	6	7	8	9
i	IF	ID	EX	MEM	WB				
i + 1		IF	ID	EX	MEM	WB			
i + 2			IF	ID	EX	MEM	WB		
i + 3				IF	ID	EX	MEM	WB	
i + 4					IF	ID	EX	MEM	WB

Tabla 2.1: Segmentacion simple de 5 etapas

Reducción de ciclos por segmentación

La segmentación proporciona la ventaja de poder lanzar una instrucción por cada ciclo de reloj. Característica que aumenta el rendimiento del procesador al obtener un menor número total de ciclos por instrucción para un mismo programa. Para conocer los ciclos por instrucción que necesita un programa se utiliza la formula:

$$\text{Ciclos por instrucción (CPI)} = \frac{\text{Número de ciclos total}}{\text{Número de instrucciones}} \tag{2.1}$$

A modo de ejemplo, veamos que sucede al ejecutar un programa de 3 instrucciones sobre un procesador que emplee 5 ciclos de reloj en ejecutar cualquier instrucción, pero en un caso no segmentado, y en otro caso segmentado en 5 etapas de 1 ciclo cada una:

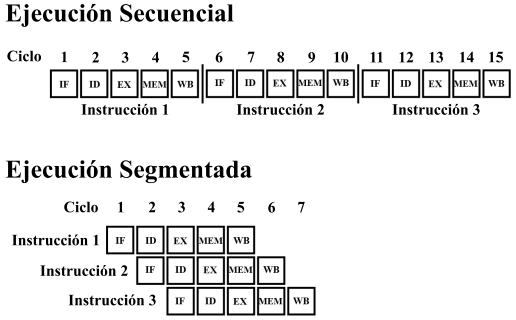

Figura 2.3: Ejecución secuencial comparada con ejecución segmentada

Como podemos ver en la figura 2.3, el procesador no segmentado tarda 15 ciclos en ejecutar las 3 instrucciones y aplicando la formula anterior se obtiene que el valor de CPI es 5. Al ejecutar el mismo programa en el procesador segmentado, este tarda 5 ciclos en ocupar las 5 etapas del procesador. A partir de ese momento con cada ciclo de reloj se completa una instrucción, completándose la ejecución del programa en 7 ciclos de reloj. El nuevo valor

de CPI es de 2,33. Así pues, la segmentación ha reducido el número de ciclos por instrucción de este programa a menos de la mitad.

Inconvenientes de la segmentación

Un programa es un conjunto de instrucciones que ejecutadas en orden realizan una tarea específica. Al permitir ejecutar una instrucción sin terminar las anteriores pueden aparecer conflictos, denominados «riesgos de segmentación» y pueden ser de los siguientes tipos: [8]

1. *Riesgos estructurales.* Surgen cuando 2 o más instrucciones necesitan acceder a los mismos recursos.

2. *Riesgos de datos.* Surgen cuando una instrucción depende del resultado de una instrucción anterior, y este todavía no se ha escrito en el registro correspondiente. A su vez pueden ser:

   - *Lectura después de escritura (RAW).* Una instrucción intenta leer un dato antes de que se escriba en el registro.
   - *Escritura después de lectura (WAR).* La *instrucción i+1* escribe el resultado en el registro antes de que la *instrucción i* haya leído el dato del mismo registro. Esto solo ocurre con instrucciones que realicen una escritura anticipada como por ejemplo, las instrucciones de auto-incremento de direccionamiento.
   - *Escritura después de escritura (WAW).* Ocurre cuando las escrituras se realizan en orden incorrecto. Por ejemplo, cuando en un mismo registro, la *instrucción i+1* escribe su resultado antes de que lo haga la *instrucción i.*
   - *Lectura después de lectura (RAR).* Realmente no es un riesgo como tal, ya que no se modifica ningún dato.

3. *Riesgos de control.* Surgen a consecuencia de las instrucciones que afectan al registro del contador de programa (PC).

2.2. ARM

La arquitectura ARM fue originalmente desarrollada por Acron Computer Limited, entre los años 1983 y 1985.

Actualmente la arquitectura ARM es el conjunto de instrucciones más ampliamente utilizado en unidades producidas. Esto se debe a su amplio uso en los sectores de telefonía móvil, sistemas de automoción, computadoras industriales y otros dispositivos.

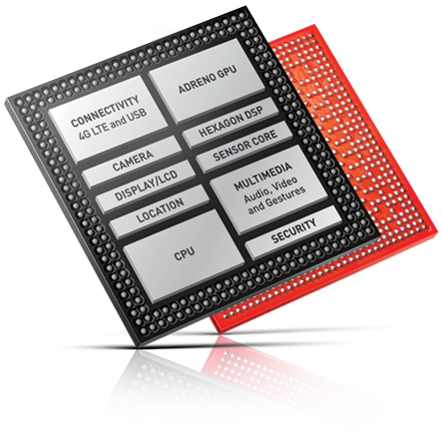

Figura 2.4: Procesador Qualcomm Snapdragon 810

Lo que hace que esta arquitectura sea tan popular es la simpleza de sus núcleos, utilizan un número relativamente pequeño de transistores, permitiendo añadir funcionalidades específicas en otras partes del mismo chip [3]. Por ejemplo, uno de los procesadores de ultima generación de la empresa Qualcomm, el «Qualcomm Snapdragon 810» esta compuesto por una CPU con 8 núcleos ARM, una unidad de procesamiento gráfico, controladores de pantalla, conectividad y cámara entre otros, todo ello en un único chip [20]. Figura 2.4.

Además de requerir poco espacio, los dispositivos ARM están diseñados con el objetivo de minimizar el consumo de energía, haciéndolo apropiado para sistemas móviles empotrados[1] que dependen de una batería.

Por último, la arquitectura ARM es altamente modular, es decir, sus componentes como pueden ser la memoria caché o los controladores, se construyen como módulos independientes y opcionales..

Todo ello no impide que la arquitectura ARM resulte muy eficiente y proporcione un alto rendimiento.

2.2.1. Arquitectura ARM

La arquitectura ARM [18] deriva de la arquitectura RISC con las características propias de esta:

- Banco de registros uniforme.
- Instrucciones de tamaño fijo.

[1] Sistema de computación diseñado para realizar una o pocas funciones dedicadas.

- Las instrucciones de procesamiento operan sobre los datos almacenados en los registros.
- Modos de direccionamiento simples.

La arquitectura ARM añade algunas características adicionales para proporcionar un equilibrio entre el rendimiento, el tamaño del código, el consumo y el silicio requerido, estas son:

- Actualización de flags en la mayoría de instrucciones.
- Ejecución condicional de instrucciones.
- Auto-incremento y auto-decremento para el direccionamiento.
- Instrucciones de carga y almacenamiento múltiple.

La arquitectura ARM contiene un banco de registros con 31 registros de propósito general. De estos sólo son visibles 16, a los cuales puede acceder cualquier instrucción. Los otros registros se utilizan para acelerar el procesado. Tres de los 31 registros tienen un uso especial y son el «puntero de pila (SP)», el «registro de enlace (LR)» y el «contador de programa (PC)».

2.2.2. Repertorio de instrucciones ARM

El repertorio de instrucciones se divide en seis categorías:

- **Salto**

  Además de permitir que las instrucciones aritmético-lógicas alteren el flujo de control, almacenando sus resultados en el registro PC, se incluye una instrucción estándar capaz de aplicar un salto de hasta 32MB hacia delante o hacia atrás.

  Otra instrucción de salto permite almacenar el valor del contador de programa en un registro para poder volver al mismo punto al finalizar el desvío. Esto es útil cuando se quiere llamar a una subrutina.

  También es posible lanzar instrucciones de salto que realizan un cambio de juego de instrucciones, en caso de necesitar lanzar subrutinas en alguno de los otros juegos de instrucciones compatibles con la arquitectura tal es el caso de Thumb o Jazelle.

- **Procesamiento de datos**

  El procesamiento de datos se realiza mediante instrucciones aritmético-lógicas, operaciones de comparación, instrucciones sobre múltiples datos, instrucciones de multiplicación y operaciones diversas.

Las instrucciones aritmético-lógicas, como su nombre describe, ejecutan operaciones aritméticas o lógicas sobre dos operandos. El primer operando siempre será un registro, mientras que el segundo puede ser un inmediato, o un segundo registro. El resultado se almacena en un registro.

Como se ha comentado anteriormente, las operaciones de comparación aplican una operación aritmético-lógica. Sin embargo no escriben el resultado en un registro, actualizan los flags de condición.

- **Transferencia de registros de estado**

  Estas instrucciones son capaces de transferir contenidos entre los registros especiales CPSR y SPSR, y los registros de propósito general.

  Al escribir en el registro CPSR se consigue establecer los valores de los bits de condición, habilitar o deshabilitar interrupciones, cambiar el estado y el modo del procesador, y cambiar el modo de acceso a memoria entre «little endian» o «big endian».

- **Carga y almacenamiento**

  Las instrucciones de carga y almacenamiento permiten transmitir datos entre los registros de propósito general y la memoria externa.

  Se pueden cargar o almacenar los registros de forma individual, un solo dato por instrucción, o de forma colectiva, un bloque de datos con una sola instrucción.

- **Co-procesador**

  Las instrucciones de co-procesador comunican el procesador principal con un co-procesador auxiliar para transmitir instrucciones o datos.

  Existen tres clases de este tipo de instrucciones: Procesado de datos, comienza el trabajo específico del co-procesador. Transferencia de instrucciones, envía o recibe datos del procesador a la memoria. Transferencia de registro, envía o recibe datos entre los registros del microprocesador y el co-procesador.

- **Excepciones**

  Las instrucciones de excepción generan interrupciones en el programa. Las instrucciones «Interrupción software» normalmente se utilizan para realizar peticiones al sistema operativo. Mientras que las instrucciones «Punto de interrupción software» generan excepciones abortando la ejecución del programa.

Los procesadores ARM son capaces de procesar instrucciones de tres repertorios diferentes. El repertorio ARM [25], el set Thumb/ Thumb-2 [1] y las instrucciones Jazelle [25].

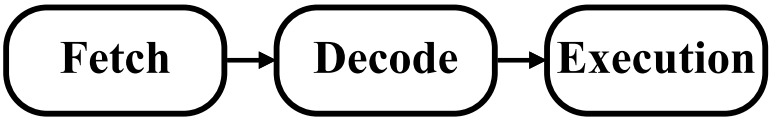

Figura 2.5: Segmentación ARM

2.2.3. Segmentación ARM

La evolución de los ARM ha significado un aumento en la cantidad de etapas en las que se divide el procesador. La familia «ARM7TDMI» consta de 3 etapas, mientras que la familia «ARM9TDMI» se divide en 5 etapas, y la familia «Cortex» se compone de 13 etapas.

La arquitectura del «ARM7TDMI», como se ha comentado, está segmentada en 3 etapas para aumentar la velocidad de flujo de entrada de las instrucciones en el procesador. Permite realizar varias operaciones al mismo tiempo y operar de forma continua. [17]

Las tres etapas en las que se divide la segmentación del «ARM7TDMI» son: (Figura 2.5)

1. **Búsqueda de instrucción**

   Se accede a la memoria para extraer la instrucción.

2. **Decodificación**

   Los registros utilizados son extraídos de la instrucción.

3. **Ejecución**

   Los valores de los registros se extraen del banco de registros, se realizan las operaciones, y se almacenan los resultados en el banco de registros.

Mientras se ejecuta una instrucción, la siguiente es decodificada y una tercera es traída de memoria.

2.2.4. Memoria ARM

Se utiliza una arquitectura Von-Neumann con un único bus de 32 bits para acceder tanto a las instrucciones como a los datos.

El único tipo de instrucciones con acceso a memoria son las instrucciones de carga y almacenamiento. Puede transmitir datos de 8, 16 o 32 bits, alineados cada 1, 2 y 4 bytes respectivamente. [17]

2.3. Field-Programmable Gate Arrays (FPGA)

Las FPGAs, del inglés Field-Programmable Gate Arrays, consisten en bloques lógicos con conexiones programables para realizar diferentes diseños

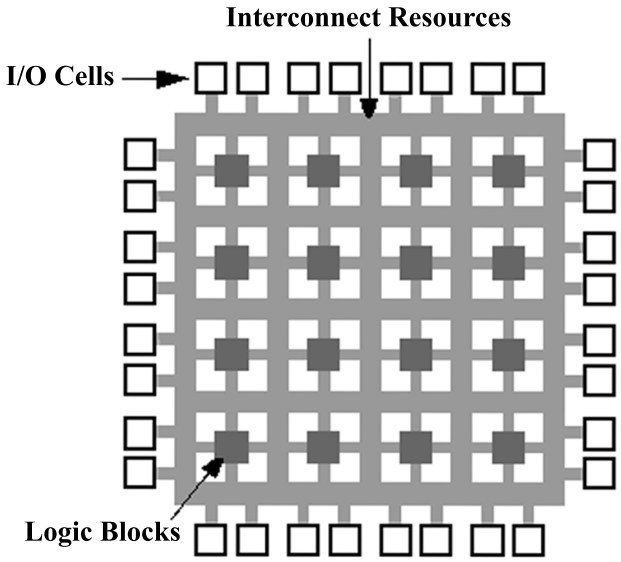

Figura 2.6: Arquitectura de una FPGA [2].

[21]. Figura 2.6.

Los bloques pueden ser tan simples como un transistor o tan complejos como un microprocesador. Las FPGAs comerciales suelen tener bloques basados en:

- Parejas de transistores.
- Puertas lógicas simples.
- Multiplexores.
- Look-up tables (LUT).
- Estructuras de puertas AND-OR.

Debido a su naturaleza re-configurable, las FPGAs conllevan un mayor coste en área, retardos y consumo de energía: requieren un área 20 veces mayor, consume 10 veces más energía y trabaja 3 veces más lenta [16]. Estas desventajas son minimizadas por la ventaja de permitir que el sistema funcione de forma casi inmediata.

Las ventajas de este tipo de dispositivos residen en su gran versatilidad, flexibilidad, su alta frecuencia de trabajo, su capacidad de procesamiento en paralelo, y a su bajo precio en comparación con los «Circuitos Integrados para Aplicaciones Específicas (ASICs)»[2].

[2]Circuitos integrados hechos a la medida para un uso particular.

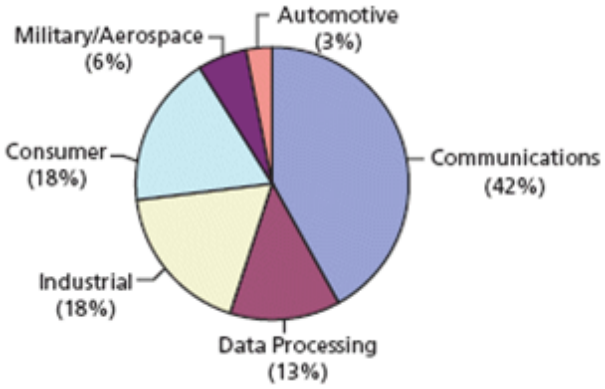

Figura 2.7: Distribución del uso de FPGAs en el año 2008.

Es por ello que las FPGAs se utilizan en todo tipo de sectores desde el procesamiento de datos y las comunicaciones hasta el sector de la automoción, el sector militar y el sector aeroespacial. Ha sido el sector de comunicaciones, como vemos en la figura 2.7, el que más uso hacía de esta tecnología en 2008.

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] 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.

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

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

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

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

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

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

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

[13] 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.

[14] 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.

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

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

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

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

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

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

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

[22] 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.

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

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

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

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

[27] 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.