Diseño, validación e implementación de una arquitectura RISC

por

Luciano César Natañe

Tesis presentada para optar al Título de

Ingeniero Electrónico

por la

Facultad de Ingeniería de la Universidad de Buenos Aires

Director:

_____

Ing. Nicolás Alvarez

Co-Director:

_____

Ing. Octavio Alpago

Miembros del Jurado:

_____

Ing. XXXXXXXXXXXXXXXXXXXX

_____

Ing. XXXXXXXXXXXXXXXXXXXX

_____

Ing. XXXXXXXXXXXXXXXXXXXX

Calificación: _____ Fecha: _____

Universidad de Buenos Aires, Facultad de Ingeniería

Diseño, valdiación e implementación de una arquitectura RISC

por

Luciano César Natale

Resúmen

El presente trabajo constituye la Tesis de Grado necesaria para obtener el título de Ingeniero Electrónico de la Facultad de Ingeniería de la Universidad de Buenos Aires.

El objetivo de este trabajo es el diseño, la validación y la implementación de una arquitectura RISC con el objetivo de generar un núcleo de procesamiento, sintetizable en FPGA, altamente configurable y suficientemente flexible y sencillo para ser utilizado en distintas aplicaciones dentro del ámbito de la investigación en los Laboratorios de Microelectrónica y de Sistemas Embebidos. Se presenta la teoría e historia necesaria para poder explicar las decisiones de diseño adoptadas en el desarrollo de la arquitectura. Se presentan vectores de prueba para las validaciones posteriores. Se verifica el diseño generado mediante emuladores. Se implementa el diseño en un lenguaje de descripción de hardware. Se sintetiza en FPGA el diseño y se contrastan los resultados con las emulaciones realizadas. Finalmente se analizan los resultados obtenidos, se contrastan los mismos contra trabajos similares, y se proponen trabajos futuros.

Agradecimientos

Agradecimientos Lucho.

Dedicatoria Lucho

Índice

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción al trabajo de tesis

El presente trabajo se encuentra enfocado en el contexto del diseño de hardware digital. El mismo fue motivado por la necesidad de contar un con núcleo de procesamiento altamente configurable y suficientemente flexible y sencillo para distintas aplicaciones dentro del ámbito de la investigación en los Laboratorios de Microelectrónica y de Sistemas Embebidos; sintetizable en FPGA[1]. La arquitectura a desarrollar será del tipo RISC[2].

Desde la aparición de los microprocesadores a mediados de los años 70, la tendencia fue el aumento de la complejidad de las arquitecturas, generando un efecto de "bola de nieve", al ir superponiendo capas sobre un núcleo central. Existió, entonces, una reacción adversa a esta tendencia. Por ejemplo, la arquitectura experimental de IBM 801; y también en Berkeley, Patterson y Ditzel fueron los primeros en acuñar el término RISC, para descibir una nueva clase de arquitectura que deshacía el camino del resto de las arquitecturas hasta el momento, conocidas, en contraposición, como CISC[3]. A partir de este antecedente, los principales fabricantes de microprocesadores han lanzado al mercado sus propias implementaciones basadas en los principios establecidos en IBM y Berkeley.

El concepto de las arquitecturas RISC se basa, principalmente, en el hecho de que al simplificar la lógica necesaria para la ejecución de una instrucción permite aumentar la frecuencia de operación de las compuertas que componen la lógica. Además, es posible

[1] "Field Programmable Gate Array" (Arreglo de compuertas lógicas reprogramables): dispositivo electrónico de lógica programable utilizado para prototipar diseños lógicos, así como también en producciones que no justifican la fabricación de millones de circuitos idénticos.

[2] "Reduced Instruction Set Computer" (Computadora de conjunto de instrucciones reducido): técnica de diseño de unidades de procesamiento basada en el hecho de que un conjunto de instrucciones simple provee una mayor performance al ser combinado con una arquitectura capaz de ejecutar dichas instrucciones en algunos pocos ciclos de máquina.

[3] "Complex Instruction Set Computer" (Computadora de conjunto de instrucciones complejo): técnica de diseño de unidades de procesamiento basadas en el hecho de que el conjunto de instrucciones debe ser lo más poderoso posible.

dividir la ejecución de las instrucciones en etapas sencillas y consecutivas, permitiendo de esta manera implementar fácilmente optimizaciones como, por ejemplo, una arquitectura de *pipeline*[4]. Es por esto que el conjunto de instrucciones es sencillo, permitiendo solamente operaciones básicas entre registros internos del microprocesador. El trabajo realizado por cada instrucción, en general, es menor que el generado por una instrucción CISC, pero se hace de manera sencilla y rápida. Es importante notar que no solamente la ganancia radica en poder aumentar la frecuencia de operación de la lógica, sino que estas condiciones facilitan el desarrollo de diseños de bajo consumo, característica muy valorada en el nicho de los sistemas embebidos.

El mercado de los sistemas embebidos es excesivamente amplio y está inserto en todas las industrias. En un automóvil, por ejemplo, podemos encontrar microprocesadores en el sistema de frenos, en la central de inyección electrónica, en el sistema de entretenimiento y navegación, etc. La otra arista de vital importancia para el mercado de los sistemas embebidos, es el de los dispositivos móviles, donde se vuelve vital el requerimiento de bajo consumo. Estamos viviendo la revolución de IoT[5], que se trata básicamente de sistemas embebidos autónomos que estan conectados a "la nube" y pueden ser monitoreados y controlados remotamente a través de Internet.

Dentro del universo de las arquitecturas RISC, actualmente se destacan dos: MIPS y ARM. La primera, fue desarrollada por un grupo de investigadores de la Universidad de Stanford (entre ellos John L. Hennessy, pionero del concepto RISC junto a David Patterson, coautores de la bibliografía más relevante del área). Esta arquitectura, por su sencillez, es la predilecta al momento del desarrollo de cursos enfocados en la enseñanza de arquitectura de computadoras. Si bien MIPS posee gran relevancia académica, es muy popular en el mercado de los microprocesadores en sistemas embebidos como equipos de telecomunicaciones, decodificadores de TV digital, y consolas de entretenimiento, con ejemplos muy conocidos como *Nintendo* y *PlaySation*. ARM, por otro lado, ha ganado una importante porción del mercado de los sistemas embebidos (con un gran aporte de los dispositivos móviles), basando su modelo de negocios en la venta de la propiedad intelectual (IP, *intellectual property*) del diseño de los microprocesadores a las empresas que finalmente producen el microprocesador.

1.1. Objetivo

La Tesis tiene como objetivo principal el diseño, la validación e implementación de una arquitectura RISC y su conjunto de instrucciones.

[4] "Pipeline": técnica de diseño de arquitecturas de computadoras en la que se segmenta la ejecución de las instrucciones en múltiples etapas, permitiendo que múltiples instrucciones estén ejecutándose en paralelo.

[5] "Internet of Things" (Internet de las cosas): es un concepto que se refiere a la interconexión digital de objetos cotidianos con internet.

El enfoque de la tesis se basará en un desarrollo teórico del conjunto de instrucciones y de las características de la arquitectura; y en el desarrollo práctico del emulador y la implementación en lenguaje descriptor de hardware.

El concepto central detrás del desarollo será el de **ortogonalidad**. Esto implica, por una parte, que los bloques constructivos de la arquitectura que se repiten sean independientes e indiferenciables entre sí. Por otra parte, los formatos de las instrucciones, en la medida de lo posible, se diseñaran de manera tal que se pueda mantener el mismo ancho de campo para los datos inmediatos y los desplazamientos (excepto en los casos donde es explícitamente conveniente agrandarlos sin penalizar la complejidad del diseño).

El objetivo perseguido va a ser el de mantener la sencillez y la ortogonalidad, favoreciendo así la simplificación de la implementación. Se trabajará en el desarrollo de la definición de la arquitectura y su conjunto de instrucciones en favor de este objetivo. Se definirá la interfaz física para la conectividad con periféricos, los tipos de datos que maneja la arquitectura, la cantidad y tipos de registros internos, el acceso a memoria de programa y de datos con su organización y modo de direccionamiento, la interfaz con la ALU[6] y la FPU[7], mecanismos de manejos de excepeciones e interrupciones, modos de operación y manejo de periféricos. Luego se definirá el conjunto de instrucciones que ejecutará la arquitectura.

Una vez definida la arquitectura y su conjunto de instrucciones, se prodecerá a diseñar los vectores de prueba para poder validar las implementaciones. Se desarrollará un emulador de la arquitectura que deberá validar los vectores de prueba diseñados. Una vez concluida esta etapa, se implementará a nivel RTL el diseño en *Verilog*. Este diseño será validado mediante simulaciones y utilizando dispositivos programables. Se validará también contra los vectores de prueba. Se analizarán los recursos utilizados en dispositivos FPGA. Se realizará un análisis comparativo entre la arquitectura desarrollada y otras arquitecturas RISC.

1.2. Alcance

Como resultados a obtener de la tesis se tienen los siguientes:

- Especificación completa de la arquitectura
- Vectores de prueba
- Emulador de la arquitectura

[6] "Arithmetic Logic Unit" (Unidad aritmético-lógica): bloque constructivo encargado de realizar las operaciones aritméticas y lógicas sobre los datos.

[7] "Floating Point Unit" (Unidad de punto flotante): bloque constructivo encargado de realizar las operaciones en punto flotante sobre los datos.

- *IP Core* codificado en el lenguaje *Verilog* de la arquitectura completa
- Resultado de los vectores de prueba tanto en el emulador como en el *IP Core*
- Análisis comparativo entre la arquitectura desarrollada y otras arquitecturas RISC
- Proposición de trabajos futuros y/o mejoras.

1.3. Organización del trabajo

En esta sección se describe la organización de la presente tesis. Con el objetivo de que la misma sea autocontenida, los primeros capítulos se ocupan de presentar las bases o conocimientos necesarios para comprender la totalidad del trabajo.

El desarrollo de la tesis se organiza de la siguiente forma:

- En el capítulo 2 se presentará la teoría general de las arquitecturas de procesadores y una revisión histórica sobre el tema. Se estudiará la diferenciación entre los universos de procesadores CISC y RISC y se justificará la elección de diseñar una arquitectura RISC para la tesis. Se presentarán las técnicas de diseño de arquitecturas estudiadas. Además se presentarán reseñas de otras arquitecturas actuales y sus decisiones de diseño, para luego contrastarlas con los objetivos perseguidos por el presente trabajo.
- En el capítulo 3 se presentará la especificación completa de la arquitectura diseñada, explicitando los criterios y las decisiones de diseño tomadas. Además se presentará el diseño de los vectores de prueba que se utilizarán para validar las implementaciones de la arquitectura.
- En el capítulo 4 se desarrollarán las implementaciones del emulador de la arquitectura y del *IP Core* en RTL. Dicho RTL cumplirá con ciertas condiciones de portabilidad y legibilidad del código, para que el mismo sea efectivamente un IP core. Se evaluará y validará el *IP Core* utilizando simuladores. Se explicitarán las decisiones de diseño necesarias para pasar de la abstracción del diseño a la implementación real.
- En el caítulo 5 se validarán las implementaciones del capítulo 4 mediante los vectores de prueba diseñados para el capítulo 3. El *IP Core* será sintetizado para distintos dispositivos FPGA. Se analizará en cada caso el consumo de recursos utilizados, máxima frecuencia de operación y la potencia consumida
- En el capítulo 6 se extraerán las conclusiones pertinentes sobre los resultados obtenidos y se propondrán futuras mejoras de la arquitecturas a partir del análisis realizado.

Capítulo 2

Marco teório: arquitectura de procesadores

En líneas generales un procesador es un sistema que permite, por un lado ingresar instrucciones y datos obteniendo en consecuencia los resultados de operar lo indicado en las instrucciones sobre los datos. No es necesario para tal fin, definir ninguna tecnología que soporte este comportamiento; se trata más bien de un desarrollo teórico. Dicho desarrollo implica distinguir las distintas partes que lo componen. En un primera aproximación, en un sistema de procesamiento podemos distinguir cuatro componentes:

- Datos de entrada
- Instrucciones
- Unidad de procesamiento
- Datos de salida

Estos componentes se relacionan de la forma mostrada en la figura 2.1. Debe notarse que los datos de entrada y las instrucciones ingresan a la unidad de procesamiento, la cual genera datos de salida como resultado de operar las instrucciones sobre los datos de entrada.
Los datos de entrada representan el dominio sobre el cual puede operar la unidad de procesamiento. Para definir un sistema de procesamiento debemos especificar, entonces, cuál es el dominio que puede manejar. Dicha definición deberá especificar el formato y cantidad de datos que acepta la unidad de procesamiento, así como los mecanismos para ingresarlos al sistema.
Las instrucciones definirán las operaciones que la unidad de procesamiento puede realizar sobre los datos de entrada. Por lo tanto, para definir un sistema de procesamiento, debemos definir qué operaciones será capaz de realizar sobre el dominio de entrada del

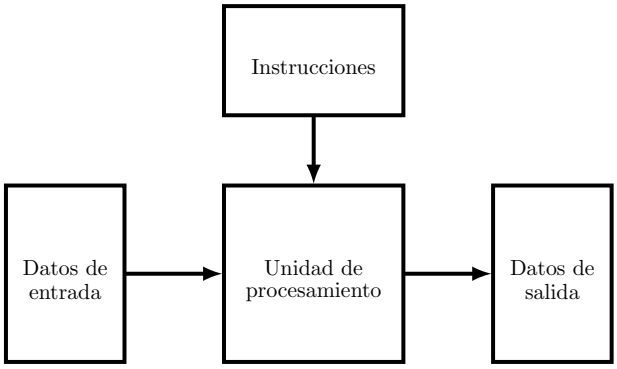


Figura 2.1: Esquema simplificado de un "sistema de procesamiento"

mismo. A su vez, se debe definir la salida esperada de las instrucciones; es por eso que al definir las instrucciones estamos definiendo intrínsecamente el dominio de salida del sistema. Es importante notar que la información sobre las instrucciones también representa una entrada para el sistema, es por eso que, como en el caso de los datos de entrada, se deberá especificar formato y cantidad que acepta la unidad de procesamiento, así como los mecanismos para ingresarlas.
Los datos de salida representan el dominio sobre el cuál las instrucciones vuelcan el resultado de las operaciones realizadas sobre los datos de entrada. El dominio de salida, como fue notado antes, queda definido al definir las instrucciones, pero debe definirse el formato y cantidad de los mismos, así como los mecanismos necesarios para extraerlos del sistema.
En este contexto la unidad de procesamiento es la encargada de recibir las instrucciones y datos, ejecutar las operaciones para finalmente presentar los resultados.
Un microprocesadores es -en efecto- una implementación de un sistema de procesamiento. El sustento físico de dicha implementación es la microelectrónica. En las siguientes secciones de este capítulo, se repasará el marco histórico en el que surgen los microprocesadores y las arquitecturas asociadas.

2.1. Perspectiva histórica

Desde épocas remotas, el hombre se ha destacado en el mundo animal por su capacidad de modificar su entorno para resolver problemas recurrentes. Es esta capacidad la fortaleza de la especie en la naturaleza. La película "2001: Odisea del esapcio"[1] narra, desde un enfoque particular, parte de la evolución del ser humano, o al menos la interpretación de los autores sobre la misma. En la misma, ubicándose temporalmente varios millones de años atrás, un clan de cavernícolas prehumanos intentan sobrevivir en condiciones

[1] "2001: Odisea en el espacio": es un film del año 1968 dirigida por Stanley Kubrick basada en la novela de Arthur C. Clarke.

extremas. Comen los pocos hierbajos que pueden encontrar en el desolado paisaje, hierbajos que para colmo han de compartir con una manada de tapires que habita la misma zona. La única fuente de agua del clan —un simple charco— les es arrebatada por un clan rival. Por si fuera poco, este desdichado clan vive permanentemente amenazado por un leopardo que domina la región y que de vez en cuando caza a alguno de sus miembros. En resumen: este grupo de homínidos padece hambre, frío y miedo, y parecen condenados a una segura extinción. En ese contexto, y por motivos que no vienen al caso, aparece uno de los cavernícolas contemplando el esqueleto de un animal. Parece reflexionar sobre lo que tiene delante, como si estuviese viéndolo desde una nueva perspectiva. Hay algo nuevo en aquellos huesos. Algo que hasta entonces ni él ni ninguno de sus congéneres habían visto. Los huesos que hay tirados por el suelo pueden ser usados. El cavernícola toma el más robusto de los huesos y empieza a golpear el esqueleto; primero con precaución, más tarde con fuerza, hasta que termina consumido por un frenesí violento. Este cavernícola acaba de descubrir el primer arma —la primera herramienta— de la historia. O dicho de otro modo, acaba de aparecer el primer ser humano sobre la faz de la tierra. Gracias al uso del hueso —o de herramientas similares como palos o piedras— el clan que estaba a punto de extinguirse descubre que puede cazar a los tapires con los que convive y comérselos. Así que sus problemas de hambre han terminado. También gracias a sus armas pueden atacar al clan rival y recuperar el charco de agua, lo que soluciona también sus problemas de sed. Y deducimos que serán capaces incluso de defenderse del peligroso leopardo. Los miembros del clan ya no son prehumanos indefensos; ahora son humanos armados.
Esta cita cinematográfica pretende graficar la diferenciación del ser humano en la cadena alimenticia. Gracias al poder de la observación y el razonamiento hemos sido capaces de modificar nuestro entorno para asegurar la supervivencia de la especie en un mundo en el que la misma se encontraba en clara desventaja. En este contexto, el ser humano ha sido capaz de generar un desarollo tecnológico, que hoy en día es vertiginoso.

2.1.1. La Máquina de Turing

Alan Mathison Turing[2] es considerado el padre de la computación y la inteligencia artificial. Sus trabajos en el campo de la computación se remontan al año 1936 cuando publica un *paper* llamado "On computable Numbers, with an Application to the Entscheidungsproblem". En este contexto, Turing inventó la llamada Máquina de Turing; esta máquina se trata de un modelo abstracto que manipula símobolos ingresados a través de un medio infinito, de acuerdo a una tabla de reglas, o para ser más exactos, es la definición del modelo matemático de una máquina capaz de hacer dicho trabajo. A pesar de la simplicidad del modelo planteado por Turing, dado cualquier algoritmo computable, una

[2]Alan Mathison Turing (23 June 1912 – 7 June 1954): típicamente considerado el padre de las ciencias de la computación y de la inteligencia artificial, fue un biólogo, matemático, lógico, científico de la computación y de la criptografía inglés que formalizó los conceptos de algoritmo y computación mediante la "Máquina de Turing", que es un modelo de una computadora de propósito general

Máquina de Turing que lo resuelva puede ser construída. Con éste modelo, reformuló los resultados hallados por Kurt Gödel en 1931 sobre los límites de prueba y computabilidad, reemplazando el lenguaje formal planteado por Gödel, por su modelo de máquina. Turing demostró que cualquier problema que pudiera ser representado por un algoritmo, podía ser resuelto por una Máquina de Turing, a su vez probando que el "problema de decisión" (entscheidungsproblem) no tenía solución, al probar que el "Halting Problem" (Problema de Parada) para las Máquinas de Turing es indeterminado según el problema de decisión. En otras palabras, para un problema de este tipo, está indeterminado el hecho de que la máquina termine de computar. Es así que la Maquina de Turing fué capaz de demostrar las limitaciones del poder de cómputo. El modelo de acceso de datos e instrucciones era secuencial, lo cuál es minimalista, pero no apto para implementaciones prácticas. La teoría de autómatas es la rama de la ciencia que hoy se ocupa de estudiar estos modelos matemáticos de computabilidad. Se establece así la propiedad de que un sistema arbitrario de instrucciones sea "Turing-Completo"

2.1.2. La revolución digital

La revolución digital es considerada la tercera revolución industrial, dando orígen a la llamada "Era de la Información". Sus comienzos se remontan a fines de los años 50. La adopción y proliferación de las computadoras digitales y el mantenimiento de registros digitales de información son las características que la definen. De manera implícita, esta revolución está relacionada con los cambios radicales provocados por la computación y las tecnologías de telecomunicaciones. En el corazón de este proceso encontramos dos componentes tecnológicas. Por un lado está la teoría, que puede remontarse a los primeros análisis matemáticos de la lógica, como el álgebra de Boole introducida por primera vez en un pequeño folleto publicado en 1847 bajo el nombre *The Mathematical Analisis of Logic* y la posterior publicación del libro *An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*, publicado en 1854. En estas publicaciónes George Boole[3] pretendió utilizar técnicas algebraicas para tratar expresiones de la lógica proposicional. En la actualidad, el álgebra de Boole se aplica de forma generalizada en el ámbito del diseño electrónico. Claude Shannon[4] fué el primero en aplicarla en el diseño de circuitos de conmutación eléctrica biestables, en 1948. Por el otro lado, se encuentra la revolución del silicio y la producción en masa y uso generalizado de circuitos lógicos digitales que permitieron el desarrollo en

[3]George Boole (2 de Noviembre de 1815 - 8 de Diciembre de 1864): Matemático, docente, filósofo y lógico Ingles. Trabajó en los campos de las ecuaciones diferenciales y el álgebra lógica. Su obra "The Laws of Thaught" (Las Leyes del Pensamiento) de 1854 describen lo que hoy conocemos como Álgebra de Boole, piedra fundamental de la "Era de la Información".

[4]Claude Elwood Shannon (30 de Abril de 1916 - 24 de Febrero de 2001): Matemático, ingeniero eléctrico y cryptografista nacido en Estados Unidos de América, es conocido como el padre de la teoría de la información. Con un *paper* publicado en 1948, describió lo que hoy conocemos como teoría de la información. Pero previamente, en 1937, también trabajó en lo que hoy conocemos como la teorída del diseño de circuitos digitales, al realizar su tesis de maestría aplicando las leyes del Álgebra de Boole a circuitos eléctricos, demostrando que podía construirse cualquier función lógica y numérica.

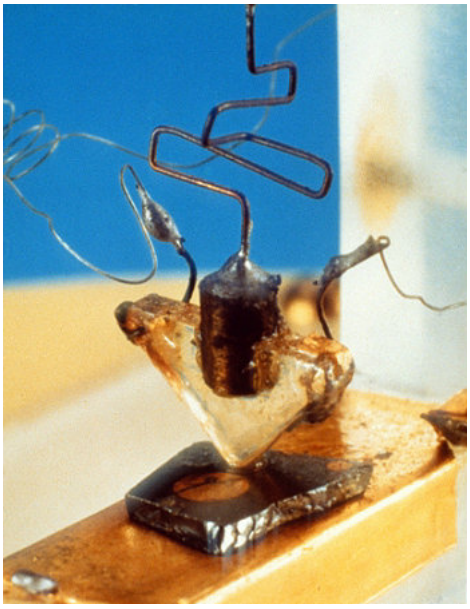


Figura 2.2: Primer transistor desarrollado por John Bardeen y Walter Brattain bajo la dirección de William Shockley en los Laboratorios Bell de AT&T en el año 1947

.

gran escala de circuitos electrónicos que implementan funciones lógicas basados en los desarrollos de Boole. Aproximadamente cien años pasaron desde que Boole publicó su teoría, hasta que la tecnología encontró el camino para implementar esos conocimientos de forma práctica y útil. La invención del transistor data del año 1947. En la figura 2.2 se muestra una imágen de una réplica de este primer transistor de la historia. Este invento fué el que permitió la creación de equipos digitales avanzados. En el contexto de la lógica digital, los transistores son utilizados como llaves de conmutación que permiten o no el paso de una señal eléctrica al ser excitados por otra señal. Previo a los transistores, la lógica era implementada con componentes electromecánicos y válvulas termoiónicas de vacío; tecnologías que por su naturaleza eran de dimensiones y consumos energéticos elevados y poco fiables. Para poner esta problemática en perspectiva, comparamos la primer computadora de propósito general electrónica, llamada ENIAC[5] implementada con 18000 válvulas, con un consumo de 160 KW, capaz de realizar 5000 sumas/s, 385 multiplicaciones/s, con 5 millones de soldaduras y un peso de 30 Ton, contra un procesador Intel Core I7 que permite 177000 MIPS con un consumo de aproximadamente 100 W, es decir, 35 millones de veces más rápido. Para hacer la misma cantidad de sumas por segundo se requerirían 5 GW con la ENIAC; considerar que la capacidad actual instalada en argentina es de 30 GW.

[5] "Electronic Numerical Integrator and Computer" (Integrador numérico y computadora electróica): Considerada la primera computadora electrónica de propósito general. Era "Touring-complete", digital y podía resolver una gran cantidad de problemas numéricos al ser reprogramable.

2.1.3. Silicio, dispositivos semiconductores y transistores

Los semiconductores son materiales, que como bien indica su nombre, no son del todo conductores. En ellos, la capacidad de conducir una corriente eléctrica puede ser manipulada de diversas maneras. En 1931 Wolfgang Pauli -quien en 1945 fue premiado con el Nobel de Física- enunció: "Uno no debería trabajar en semiconductores, eso es un lío deleznable, quién sabe si realmente existen". Nada más alejado de la realidad que hoy nos rodea. Los materiales semiconductores permitieron la creación de los transistores y posteriormente los circuitos integrados, dispositivos que iniciaron la revolución digital. Los primeros dispositivos fueron fabricados sobre germanio y arsenurio de galio. Incluso, el primer circuito integrado, fue realizado en germanio. Pero fue el silicio el material que realmente revolucionó la industria, por su alta disponibilidad en la naturaleza y relativa fácil manipulación para la fabricación de dispositivos semiconductores en circuitos integrados. Puede afirmarse que el silicio es uno de los materiales mejor conocidos por el ser humano. Hace más de 50 años que se lo estudia en detalle para mejorar la tecnología, logrando importantes avances. También puede afirmarse que el los transistores hoy en día es el bien más abundante en el mundo. Una comparativa del año 2012 muestra que durante año se produjeron en el órden de 10^{17} granos de arroz en la tierra, mientras que en el mismo período en se produjeron en el órden de 10^{19} transistores, según declaraciones de la *Semiconductor Industry Asociation* de los Estados Unidos de América. La clave de semejante número en la producción son los circuitos integrados.

2.1.4. Circuitos integrados y tencología CMOS

En la figura 2.3 puede verse el primer circuito integrado de la historia. Medía aproximadamente media pulgada de ancho e implementaba dos transistores montados en una barra de germanio. Nace así el concepto del *chip* o *microchip*: en inglés, *chip* significa "corte o fracción pequeño de un material duro" directamente relacionado con la técnica de fabricación de circuitos integrados donde, a grandes rasgos, una barra cilíndrica de cristal de silicio puro, es cortada en muy delgadas láminas en forma de discos, sobre las cuales se "imprimen" los circuitos integrados, repitiendo el mismo patrón múltiples veces en un mismo disco para finalmente cortar ese disco en diminutos fragmentos que contienen el diseño. La evolución de las técnicas de fabricación permitieron integrar en un mismo *chip* de silicio más de un dispositivo, permitiendo implementar circuitos relativamente complejos en una pequeña área de silicio, disminuyendo así los riesgos de fallas por interconexión entre dispositivos. A su vez, las técnicas de fabricación evolucionaron permitiendo escalar el tamaño de los dispositivos fabricados incrementando la cuenta de dispositivos integrados por unidad de área. Como consecuencia buscada de disminuir el tamaño de los dispositivos, se logró aumentar la velocidad máxima de conmutación que los mismos pueden lograr, redundando en generar lógica cada vez más compleja y rápida en un mismo *chip*. En simultáneo con estos avances se comenzaron a fabricar transistores

MOSFET[6], transistores de efecto de campo eléctrico, que como gran ventaja sobre los transistores bipolares de juntura, evitaban la disipación de potencia al mantenerse en un estado definido (encendido o apagado), es decir que sólo disipaban potencia al cambiar de estado. Es así que en el año 1978 aparece la tecnología CMOS[7] la cual permitió elevar el nivel de integración en forma masiva, manteniendo bajos niveles de disipación de potencia. Debido a la relativa sencillez geométrica del diseño de los transistores MOS, se pueden reutilizar diseños escalándolos para las nuevas generaciones de tecnología. El 19 de abril de 1965 Gordon Moore[8], cofundador de Intel Corporation[9], estableció de forma empírica que la cantidad de dispositivos integrados en un circuito integrado se duplicaría cada año. Más tarde, en 1975, modificó su propia ley al corroborar que el ritmo bajaría, y que la capacidad de integración no se duplicaría cada 12 meses sino cada 24 meses aproximadamente. Esta progresión de crecimiento exponencial, duplicar la capacidad de los circuitos integrados cada dos años, es lo que se denomina ley de Moore. Sin embargo, en 2007 el propio Moore determinó una fecha de caducidad: "Mi ley dejará de cumplirse dentro de 10 o 15 años", no obstante también aseveró que una nueva tecnología vendrá a suplir a la actual. El cumplimiento se ha podido constatar hasta hoy.

La ley de Moore, no es una ley en el sentido científico, sino más bien una observación del ritmo de avance de la industria de aquellos momentos. Al momento de publicar esas declaraciones, Moore trabajaba en los Laboratorios de Fairchild Semiconductor, donde trabajaba junto a Robert Noyce. Ellos fueron los fundadores de Intel en 1968. El ritmo de crecimiento de la insdustria de los Semiconductores dió lugar así, entre otras cosas, a la creación de los microprocesadores.

2.1.5. Microprocesadores

Los microprocesadores surgen como consecuencia del alto nivel de integración y complejidad que la tencología de circuitos integrados permitieron alcanzar. Una computadora digital, podía ser construída en uno o algunos pocos *chips* de circuitos integrados. Los primeros diseños de microprocesadores propiamente dichos datan de fines de los años

[6] "Metal-Oxide-Semiconductor Field Effect Transistor" (Transistor de efecto de campo Metal-Óxido-Semiconductor): Es un tipo de transistor fabricado en silicio, donde se genera una estructura que superpone una capa metálica sobre un óxido sobre material semiconductor, generando una estructura capacitiva, con la habilidad de manejar la presencia de un canal conductivo al inducir un campo eléctrico en el material semiconductor. Éste tipo de transistores presenta diversas ventajas frente a los transistores bipolares de juntura para el funcionamiento en circuitos digitales.

[7] "CMOS": técnica de diseño de circuitos lógicos, que utiliza sólo transistores MOSFET-N y MOSFET-P para implementar cualquier función lógica.

[8] Gordon Earle Moore (Nacido el 3 de Enero de 1929): co-fundador de Intel Corporation, es el autor de la "Ley de Moore".

[9] Intel Corporation: fundada el 18 de Julio de 1968 por Goordon Moore y Robert Noyce, es hoy en día la empresea que hace punta en el diseño de circuitos integrados y tecnologías de fabricación. En sus comienzos, su principal negocio fue el desarrollo de circuitos de memorias SRAM y DRAM, pero a pesar de ser la empresa que creó el primer microprocesador comercial en 1971, fué luego de la aparición de las "PC" que el diseño de microprocesadores se volvió su principal negocio.

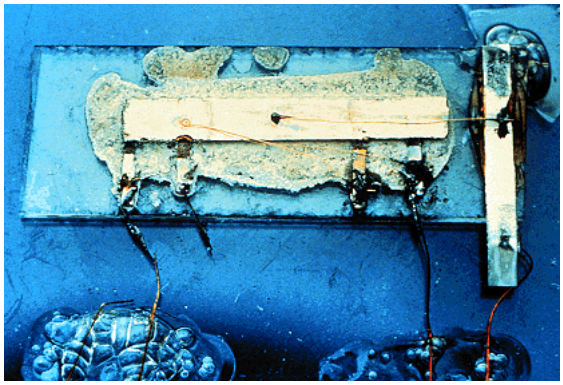


Figura 2.3: Primer circuito integrado presentado por Texas Instruments, Inc. el 12 de Septiembre de 1958

60. Dentro de este contexto aparece el término *Central Processing Unit* (Unidad Central de Procesamiento) o CPU, que es la parte de la máquina encargada de tomar los datos de entrada y las intrucciones y generar los resultados, dentro del esquema del sistema de procesamiento planteado al principio de éste capítulo. El microprocesador integra además otros componentes y funcionalidades y se vale de ciertos periféricos que pueden estar o no integrados en el mismo circuito, como por ejemplo, memorias de sólo lectura (*Read Only Memory* o ROM) y memorias de acceso aleatorio (Random Access Memories o RAM) y dispositivos de entrada / salida (*I/O, input output devices*. El diseño de arquitecturas de microprocesarores, como todo proceso de innovación tecnológica, no estuvo exento de discusiones y distintas vertientes y prácticas a seguir. Los primeros diseños de los que se tenga registro en órden cronológico, fueron:

CADC

En 1968, la armada de los Estados Unidos de América le encomienda a la empresa Garret AiResearch[10] la producción de una computadora digital que pudiera competir con los sistemas electromecánicos que estaban bajo desarrollo para los sistemas de control de vuelo del nuevo avión de combate F-14 Tomcat. El diseño fue completado en 1970 y utilizaba un conjunto de chips (*chipset*) MOS como CPU, siendo aproximadamente 20 veces más pequeño que su contraparte electromecánica y a su vez, mucho más confiable. Este diseño fue utilizado en todos los primeros modelos de Tomcat. La armada se rehusó a publicar el diseño hasta 1997, es por eso que la CADC (Central Air Data Computer) y su *chipset* asociado MP944 no son muy conocidos y no son considerados en la mayoría de la bibliografía reelevante y es a partir de ese momento en que son reconocidos como los primeros diseños válidos de microprocesadores, y no sólo eso, si no que también son

[10]Garret AiResearch: fundada en 1936 en Los Ángeles por John Clifford "Cliff" Garret, fue una compañía dedicada al desarrollo de tecnologías relacionadas a la industria militar aérea.

Figura 2.4: Grumman F14 Tomcat, avión de combate de la US Navy cuyos sistemas de vuelo eran controlados por la CADC.

la primera especie de *Digital Signal Processor* (Procesador de Señales Digitales) o DSP[11] puesto que además de las funciones de microprocesador, implementaba la funcionalidad de medir variables como altitud, velocidad vertical y número de *mach* a partir de datos de sensores pitot, presión estática y temperatura. Los responsables del diseño fueron Ray Holt y Steve Geller.

Four-Phase Systems AL1

Un diseño de Lee Boysel que data del año 1969 dentro de Texas Instruments[12], era un componente de una implementación en partes de un microprocesador de 24 bits, compuesto por tres chips iguales: AL1. Durante un juicio por patentes, se demostró que una sóla AL1 junto con memorias ROM y RAM e I/O era capaz de funcionar como CPU.

Pico/General Instruments

En 1971, Pico y General Instruments (GI) colaboraron en el diseño de circuitos integrados con el objetivo de fabricar una implementación de un diseño de *chip* único para la calculadora Monroe/Litton Royal Digital III Calcultator. Integraba en el mismo *chip*, memoria ROM y RAM llamado PICO1/GI250. Pico era un emprendimiento de cinco ingenieros de diseño de GI, que tenían la visión de crear esta arquitectura en un único *chip*. Contaban con experiencia previa en diseños de *chipsets* tanto de GI como de Marconi-Elliot. Algunos de ellos habían trabajado para Elliot Automation para crear una computadora de 8 bits en tecnología MOS, y habían ayudado a establecer un laboratorio

[11] "Digital Signal Processor" (Procesaro de señales digitales): es un tipo de microprocesador especializado en el tratamiento digital de señales

[12] Nota sobre TI

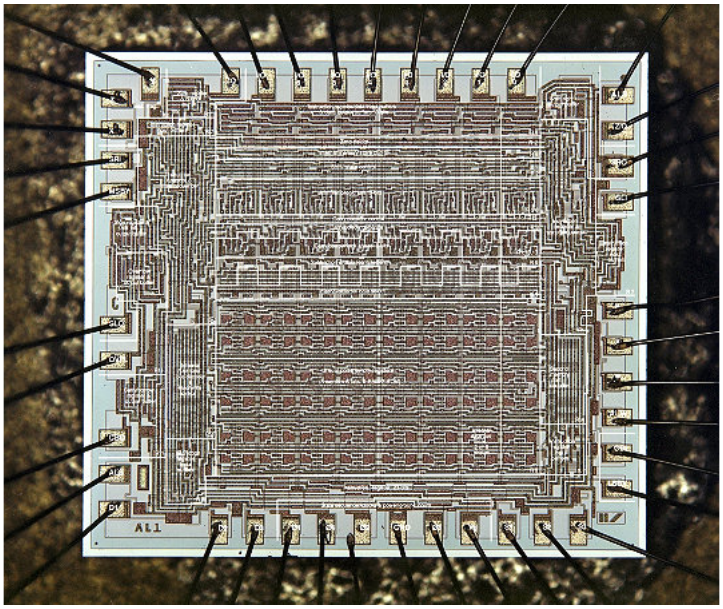


Figura 2.5: Imágen microscópica de la AL1 desarrollada por Four-Phase Systems Inc..

de investigación en tecnologías MOS en Escocia en el año 1967. El mercado de las calculadoras estaba en pleno auge, con lo cuál estos diseños supusieron un éxito comercial para Pico y GI. GI, por su parte continuó con la innovación en microprocesadores y microcontroladores y la división GI Microelectronics se convirtió en 1987 en Microchip PIC Microcontroller.

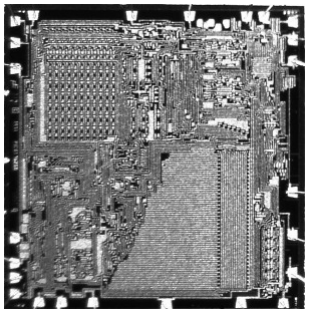


Figura 2.6: Imágen microscópica del PICO1/GI250- Desarrollo colaborativo entre Pico y General Instruments de *chip* único para la Monroe/Litton Royal Digital III Calcultator.

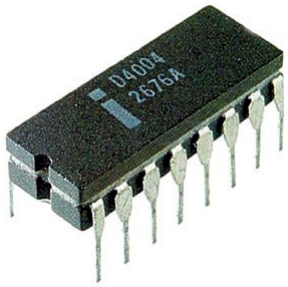


Figura 2.7: Encapsulado del Intel 4004; que es considerado el primer microprocesador comercial en ser introducido en el mercado.

Intel 4004

El Intel 4004 es generalmente el reconocido como el primer microprocesador comercial disponible en el mercado. El primer anuncio respecto de este dispositivo data del 15 de Noviembre de 1971, en la publicación Electronic News. El diseño nace a partir del requerimiento de una compañía japonesa fabricante de calculadoras llamado Busicom, que le encomienda a Intel la tarea de desarrollar un *chipset* para calculadoras de escritorio de alto rendimiento. El diseño original requerido por Busicom especificaba un *chipset* compuesto por 7 *chips* diferentes para la realización de una CPU de propósito específico cuyo programa estuviera almacenado en una ROM y sus datos guardados memoria de lectura y escritura implementada con registros de desplazamiento. Intel le asignó el proyecto a Ted Hoff, quien propuso simplificar el diseño utilizando almacenamiento en memoria RAM dinámica y una arquitectura de CPU de propósito general. El diseño propuesto por Hoff implementaba la solución en 4 *chips*: uno de ROM para almacenar el programa, uno de RAM dinámica para almacenar los datos, un dispositivo sencillo de I/O y una CPU de 4 bits. A pesar de no ser específicamente un diseñador de circuitos integrados, el supuso que se podía integrar todo el CPU en un único *chip*. Estas especificaciones surgieron de la interacción de Hoff con un empleado a su cargo, el ingeniero de software llamado Stanley Mazor y un ingeniero de Busicom llamado Masatoshi Shima. Este proyecto se llamó MCS-4 y no fué hasta que Intel contratara al italiano Federico Faggin que empezó a tomar su forma definitiva. Faggin venía de desarollar en 1968 en Fairchild Semiconductor una teconología de compuertas de silicio (SGT o *Silicon Gate Technology*), técnica que se sigue aplicando hoy en día. Faggin también era responsable del desarrollo de la técnica llamada *Random Logic* que permitió la síntesis de descripciones complejas de lógica en hardware sencillo, como compuertas AND y OR. En el momento en el que se estaba desarrollando el proyecto MCS-4, el responsable del área de diseño de MOS de Intel era Leslie L. Vadász, cuya atención estaba enfocada en el mercado de memorias. Fue gracias a esto que le otorgó el liderazgo del proyecto MCS-4 a Faggin, quien fue finalmente el responsable de llevar el proyecto hasta la concepción final del 4004 gracias a la aplicación de las técnicas de diseño y fabricación antes mencionadas. Las primeras unidades del 4004 fueron entregadas a Busicom en Marzo de 1971.

2.2. Evolución hacia las arquitecturas modernas

El concepto moderno de las arquitecturas se basa en las máquinas que almacenan tanto el programa que se ejecuta, como los datos que se manjan en memoria de lectura-escritura (*Stored-Program digital computer*). Esto las diferencia de las primeras implementaciones de los años 40, tales como Colossus[13] y la ENIAC.
En esta sección se estudian las distintas clasificaciones de arquitecturas existentes hoy en día. Luego se detallarán diversas arquitecturas existentes y se las enmarcará dentro de dichas clasificaciones.

2.2.1. Clasificación de las arquitecturas según el acceso a la memoria de instrucciones y datos

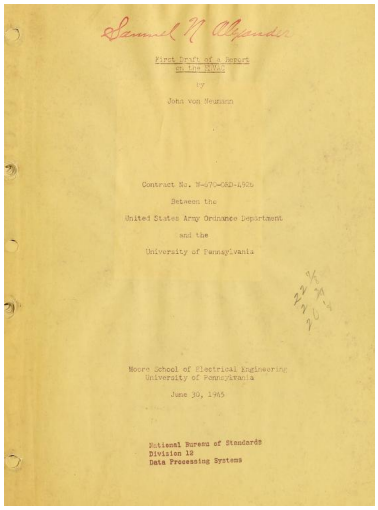
Las instrucciones que ejecuta el procesador, al igual que los datos del programa en ejecución, deben residir en la memoria. De este hecho se desprende que existen -al menos- dos tipos de memorias: de instrucciones, y de datos. El acceso a estas dos memorias puede realizarse a través del mismo *bus*[14] o con *buses* separados. Los nombres que la historia les ha provisto a estos dos posibles enfoques, son *von Neumann* para las arquitecturas de *bus* único y *Harvard* para aquellas de *buses* separados.

Arquitectura von Neumann

También conocida como modelo de von Neumann y arquitectura Princeton, este tipo de arquitectura debe su nombre a John von Neumann[15], quién en el año 1945 describió en el documento inconcluso, cuya carátula puede verse en la figura 2.8 y conocido como *First Draft of a Report on the EDVAC*[1], una arquitectura de computadora electrónica digital para ser implementada con válvulas de vacío. Ésta fue la primera publicación dónde se describe el diseño lógico de una computadora que utilizase el concepto de *stored-program* o programa almacenado. Este diseño, tal como fue llamado por el propio von Neumann, *very high speed automatic digital computing system* o sistema automático digital de cómputo de muy alta velocidad, estaba dividido en 6 componentes:

- CA: *central arithmetic* o aritmética central
- CC: *central control* o control central
- M: *memory* o memoria

[13] Colossus era el nombre de una serie de computadoras desarrolladas por los británicos para desencriptar las comunicaciones enemigas durante la segunda guerra mundial. Implementadas con válvulas termoiónicas, resolvían mediante lógica booleana operaciones de cálculo.
[14] Bus: definir bus
[15] Nota sobre el chango


Samuel N. Alexander

First Draft of a Report
on the EDVAC

by

John von Neumann

Contract No. W-670-ORD-4926

Between the

United States Army Ordnance Department

and the

University of Pennsylvania

Moore School of Electrical Engineering
University of Pennsylvania

June 30, 1945

National Bureau of Standards
Division 12
Data Processing Systems

Figura 2.8: Carátula del escrito de von Neumann First Draft of a Report on the EDVAC

- I: *input* o entrada
- O: *output* o salida
- R: *external memory* o memoria externa

La memoria almacena tanto números (datos) como órdenes (instrucciones). La aritmética central podía realizar sumas, restas, multiplicaciones, divisiones y raices cuadradas. Otras operaciones matemáticas, como logaritmos y funciones trigonométricas se debían realizar utilizando tablas de búsqueda e interpolaciones, posiblemente bicuadráticas. Una decisión de diseño establecida en el documento estableció que multiplicaciones y divisiones podían realizarse mediante tablas logarítmicas, pero para poder mantener dichas tablas pequeñas debería utilizarse interpolaciones, lo cuál a su vez necesita de multiplicaciones, aunque de menor precisión.
Los números se represetaban mediante notación binaria. Estimó que 27 dígitos binarios[16] deberían ser suficientes, pero redondéo a 30 dígitos, más uno de signo y otro para diferenciar los números de las órdenes, resultando en palabras de 32 dígitos binarios. La aritmética utilizada era complemento a dos para simplificar la operación de resta. Para la multiplicación y la división, propuso ubicar el punto binario luego del bit de signo, lo que implica que el dominio de los operandos y resultados está en el rango -1 a 1, y por lo tanto, los datos y resultados de los programas deberían ser escalados acordemente.
En cuanto al diseño de los circuitos, estableció que deberían utilizarse válvulas de vacío, dejando de lado los relé, dada a la mayor velocidad provista por las primeras. Otras sugerencias involucraban mantener al sistema de cómputo lo más sencillo posible, evitando

[16]En el documento se hace meción a dígitos binarios y no a *bits*, término que fue acuñado en 1948 por Claude Shannon

cualquier tipo de optimización de performance mediante la superposición de operaciones. Las operaciones aritméticas debían realizarse de a un dígito binario a la vez. Estimó el tiempo de la suma de dos dígitos binarios en un microsegundo, por lo tanto la multiplicación de dos números representados por 30 dígitos binarios deberí realizarse en aproximadamente 30^{20} microsegundos; lo cuál era mucho más rápido que el tiempo de cualquier dispositivo de cálculo del momento. El diseño estaba basado en lo que von Neumann llamó "elemento E", basándose en el modelo biológico de las neuronas, pero implementado de forma digital planteando que podrían ser fabricados mediante una o dos válvulas. En términos modernos, la forma más sencilla del "elemento E" es una compuerta *AND* de dos entradas, con una de sus entradas invertidas, llamada "entrada de inhibición". Al agregar más entradas a este dispositivo, se establecía un nivel de umbral, el cual al ser superado por la suma de una determinada cantidad de entradas generaba una salida en tanto y en cuanto no se excitara la entrada de inhibición. También estableció que dichos elementos de múltiples entradas podían ser construidos utilizando combinaciones de la versión elemental, pero recomendaba implementarlos por completo utilizando así menos válvulas de vacío con el objetivo de lograr circuitos más sencillos y rápidos, principio que hoy en día se mantiene vigente.
Toda función lógica arbitraria, podía implentarse, entonces, a partir de dichos "elementos E". Demostró en este trabajo, cómo implementar circuitos que implementaban las funciones aritméticas, así como elementos de memoria y circuitos de control, sin referirse en ningún momento al término de "lógica binaria".
Los ciruitos debían ser sincrónicos, obteniendo la señal de reloj a partir de un circuito oscilador implementado con válvulas y posiblemente controlado mediante cristales osciladores. Los diagramas lógicos incluían los tiempos de demora representados mediante una flecha. Estimó que la velocidad a la que se movía un pulso eléctrico en un cable era de $300 mts/microsegundo$, por lo tanto no debería generar problemas hasta obtener velocidades de reloj de $100 MHz$. Tambíen menciona pero no desarrolla la necesidad de contar con mecanismos de detección y correción.
En cuanto a la memoria, que es donde entra la clasificación discutida en esta sección, von Neumann estableció que la memoria es uniforme, conteniendo tanto los números (datos) como las órdenes (instrucciones). Citando su trabajo:
"El dispositivo requiere una memoria considerable. A pesar de que pareciese que distintas partes de la memoria tiene que realizar funciones que difieren en su naturaleza y considerablemente en su propósito, resulta tentador tratar a toda la memoria con un sólo órgano, y hacer que sus partes sean tan intercambiables como sea posible para todas las funciones que deban realizar" [1, sección, 2.5]
"Lás órdenes recibidas por CC vienen de M, es decir, el mismo lugar donde se almacenan los datos numéricos"[1, sección, 14.0].
Él concluyó que la memoria sería el subsistema más grande y abarcativo de la máquina. Basándose en diversos problemas matemáticos, incluyendo la resolución de ecuaciones diferenciales parciales y ordinarias, ordenamientos y experimentos probabilísticos, estimó que se necesitaba espacio para almacenar 8192 palabras de 32 dígitos binarios para los datos, y algunos cientos de palabras para almacenar las órdenes. Para la implementa-

ción de memoria, propuso dos tipos de memoria rápida, *delay line memory* e *iconoscope*. Con estas implementaciones planteó que la memoria debía ser direccionable por palabras y para sortear los problemas de demora asociados a la lectura de estas memorias, organizó las mismas en 256 conjuntos de 1024 dígitos binarios, o sea 32 palabras, logrando así direccionar los conjuntos con 8 dígitos binarios y las palabras con 5 utilizando, entonces, 13 dígitos binarios en total para el direccionamiento completo.
En su trabajo también estableció el formato de las órdenes, al cual llamó "código". Los tipos de órdenes incuyeron las operaciones aritméticas básicas, así como el movimiento de palabras entre CA y M (análogas a las instrucciones *load* y *store* de hoy en día que veremos más adelante), una órden que elegía entre dos números basado en el signo del resultado de una operación previa (análoga a una instrucción *branch*), órdenes para controlar la entrada y salida de datos y para indicarle a la CC que debía tomar instrucciones desde otra sección de M (análoga a un *jump*). Determinó, asimismo, la cantidad de dígitos binarios necesarios para necesarios para los distintos tipos de órdenes, sugirió lo que llamó "órdenes inmediatas", donde la siguiente palabra se trata del operando (lo cual también tiene una anagolía con ciertos conjuntos de instrucciones actuales) y dejó planteado si era deseable dejar dígitos sin especificar para propósitos futuros y mayor direccionamiento de memoria.

Arquitectura Harvard

Este tipo de arquitectura, ve sus orígenes en un desarrollo propuesto por el Dr. Howard Aiken[17] en la universidad de Harvard en el año 1937 basado en el trabajo llamado *Analytical Engine*[18] de Charles Babage[19]. El desarrollo fue llevado al cabo por IBM[20] y se trataba de una máquina de cálculo llamada *Automatic Sequence Controlled Calculator - Harvard Mark I* entregado a la universidad el 24 de agosto de 1944. A diferencia de la máquina propuesta (posteriormente) por von Neumann, ésta era una máquina electromecánica construida con llaves, relés, engranajes y embreagues. Constaba de 765000 componentes electromecánicos, cientos de millas de cableado, un peso de 4.5 Ton. y un consumo de potencia de 3.7 kW. Para el ingreso de datos, contaba con 60 conjuntos de 24 llaves rotativas que permitían ingresar números decimales. Podía almacenar hast 72 números de hasta 23 dígitos decimales. En un segundo era capaz de realizar 3 sumas o restas. La multiplicación tardaba 6 segundos y una división 13.5 segundos. El calculo de funciones logarítmicas o trigonométricas utilizaba más de un minuto. Las instrucciones eran ingresadas a través de cintas perforadas de 24 canales (en la figura 2.9 se puede ver una cinta con instrucciones de esta máquina). Otra cinta podía ser utilizada para el ingreso de datos, pero utilizando un formato distinto al de las instrucciones; estas no podían ser ejecutadas desde los registros de almacenamiento. Esta separación entre datos e instrucciones es lo que se conoce como arquitectura Harvard y que la diferencia de la

[17]Nota sobre Aiken
[18]Nota sobre Analytical Engine
[19]Nota sobre el tipo
[20]Nota sobre IBM

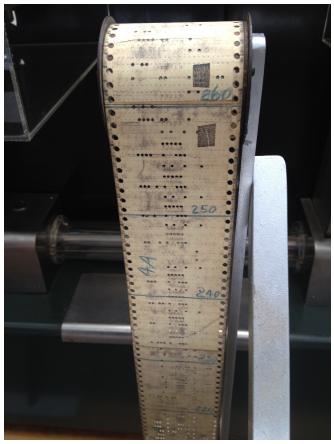


Figura 2.9: Cinta de papel perforado con instrucciones de la Mark I

arquitectura von Neumann. El formato de las instrucciones definía tres campos en ocho canales separados. Cada acumulador, cada conjunto de switches, y los registros asociados a la entrada y salida y las unidades aritméticas tenían asignado un índice único. Estos números eran representados en formato binario en la cinta perforada de control, siendo el primer campo el índice del resultado de la operación, el segundo el orígen del dato para la operación y el tercero el código que representa la operación a realizar. Una particularidad de ésta primera versión es que no tenía instrucción para el salto condicional en el programa. Por lo tanto, los programas complejos debían ser largos y los "loops" se implementaban uniendo físicamente el final de la cinta perforada del programa con el principio.

La Mark I fue sucedida por la Mark II en el año 1948, la Mark III en septiembre de 1949 y la Mark IV en el año 1952, todos estos proyectos fueron de Aiken. La Mark II fue una mejora sobre la Mark I, pero aún basada en componentes electromecánicos. La tercer versión utilizaba en su mayoría componentes electrónicos tales como válvulas de vacío y diodos cristalinos, pero utilizaba cilindros magnéticos para almacenamiento, y relés para la transferencia de datos entre ellos. La cuarta versión fue la primera en ser completamente electrónica, reemplazando los cilindros magnéticos de almacenamiento con memorias de núcle magnético; el tipo de memoria de acceso aleatorio predominante entre 1955 y 1975.

Harvard vs. von Neumann y Harvard modificada

Como fue establecido al principio de esta sección, lo que diferencia estas dos arquitecturas es el acceso a los datos y a las instrucciones. La arquitectura von Neumann puede tratar a los datos como instrucciones y viceversa, dado que para dicha arquitectura, la memoria es exactamente la misma, con el mismo tipo de acceso y el mismo direccionamiento, contrariamente a lo que es una arquitectura puramente Harvard, cuyas

memorias de datos e instrucciones se acceden por diferentes medios y cuyas direcciones pueden estar sobrepuestas (es decir que la misma dirección de memoria no representa lo mismo si se trata de datos o intrucciones) y por lo tanto, los datos y las intrucciones no pueden ser intercambiados en el tratamiento que se realiza dentro de la unidad de cómputo. Dadas estas condiciones, las principales desventajas son que, la arquitectura von Neumann no puede acceder simultáneamente a datos y a instrucciones y la máquina de Harvard no puede ni leer ni escribir en el espacio de memoria de las instrucciones, denegando por lo tanto cualquier posibilidad de código auto-optimizable o inteligente que reescriba sus instrucciones, así como tambíen la carga de los programas a partir de medios de almacenamiento de datos.
Hoy en día, las máquinas puramente Harvard no son más que una especialidad. Típicamente, en las máquinas modernas, se implementas arquitecturas Harvard modificadas. Estas modificaciones pueden ser de diversos tipos:

Arquitectura de caché separado Es la modificación más común sobre la arquitectura Harvard, se construye una jerarquía de memoria donde el nivel más cercano al núcleo de procesamiento (típicamente llamado caché de nivel 1) está separado en datos e instrucciones, y unificando estas memorias en un nivel de jerarquía superior. Con esta técnia, se puede relajar el problema de acceder a la memoria de instrucciones como datos, tanto para la lectura como para la escritura al unificar toda la memoria en un solo espacio de direccionamiento, emulando así el comportamiento de una arquitectura de von Neumann, pero manteniendo la ventaja de poder acceder a datos e instrucciones de forma simultánea. Esta característica será transparente para la mayoría de los programadores, excepto aquellos casos en los que se necesario el manejo de técnicas como por ejemplo, la coherencia de caches.

Acesso a instrucciones como datos Un arquitectura que provea esta solución matiene la naturaleza de espacios de direcciones separados de la arquitectura Harvard, pero incluye operaciones especializadas en acceder al contenido de la memoria de instrucciones como si fueran datos. Como los datos no pueden ser ejecutados driectamente como instrucciones, estas implementaciones no siempre son vistas como Harvard "modificadas".
Esta técnica puede presentarse con dos variantes. Por un lado, el acceso puede ser de sólo lectura, provee la capacidad de copiar contenido embebido en el código cargado en la memoria de instrucciones a la memoria de datos cuando el programa comienza o, mejor aún, si los datos no van a cambiar (como es el caso de constantes aritméticas o cadenas de texto predefinidas), estos datos que están en la memoria de instrucciones pueden ser directamente leídos en tiempo de ejecución por el programa sin tomar espacio de la memoria de datos (que puede ser escasa en las implementaciones que aplican esta técnica). Por otro lado, dicho acceso puede ser de lectura/escritura, suele darse cuando es necesaria una capacidad de reprogramación de la máquina en cuestión. Pocas computadoras hoy en día están basadas únicamente en ROM para su funcionamiento. Puede ser el

caso de algunos microcontroladores que tienen operaciones para escribir en su memoria "Flash" (en la cual se almacenan las instrucciones). Esta capacidad provee medios de acceso para actualizaciones de software/firmware y reemplaza a las EEPROM.

Ejecucion de datos Algunas arquitecturas pueden obtener sus instrucciones desde cualquier segmento de memoria con la limitación de que no podrá acceder simultáneamente para leer instrucciones y datos de un mismo segmento de memoria. Para lograr este comportamiento, se direccionan los buses tanto de datos como de instrucciones sobre distintos segmentos físicos de una misma memoria.

Características de las arquitecturas Harvard modificadas Tres características pueden ser utilizadas para diferenciar a este tipo de arquitectura de las Harvard puras y de las von Neumann:

Las memorias de datos e instrucciones ocupan diferentes espacios de direccionamiento Para las máquinas puramente Harvard, hay una dirección "cero" de instrucciones en el espacio de direcciones de las instrucciones y una dirección "cero" de datos en el espacio de direcciones de los datos que referencian a un lugares de almacenamiento distintos. En contraste, las máquinas von Neumann y algunas Harvard modificadas (como las de caché separado), almacenan tanto datos como instrucciones en un espacio de direcciones unificado, por lo tanto, la dirección "cero" hace referencia a una sóla cosa y lo almacenado allí puede ser la representación binaria de un código de instrucción o de un dato, dependiendo de cómo fue escrito el programa en ejecución. Sin embargo, tal como las máquinas puramente Harvard, las Harvard modificadas con espacios de direcciones separados, pero con instrucciones específicas que permiten leer y escribir las instrucciones como datos, tienen una direcciónes solapadas para los respectivos espacios de direccionamiento, por lo tanto esto no distingue las máquinas puramente Harvard de este último tipo de Harvard modificadas.

Las memorias de datos e instrucciones tienen conexiones físicas separadas hacia el núcleo de procesamiento Este es el motivo principal por el que existen las arquitecturas Harvard (tanto puras como modificadas), y por qué coexisten con las más generales y flexibles von Neumann: los caminos separados físicamente para los datos y las instrucciones permiten que las instrucciones sean cargadas en la unidad de cómputo al mismo tiempo que los datos, aumentando así el rendimiento. Las máquinas puramente Harvard tienen caminos separados con espacios de direcciones separados. Las máquinas de caché separado proveen este acceso separado para la unidad de procesamiento pero presentan un espacio de direccionamiento unificado al ascender en la jerarquía de memoria. Una máquina von Neumann puede tenera sólamente un espacio de direcciones unificado. Para el punto de vista del programador, una arquitectura de cache separado es

tratada de forma equivalente a una von Neumann (al menos hasta el punto en el que el tratamiendo de coherencia de cache se convierta en significativo), como puede ser el caso de código auto-optimizable o la carga de un programa en memoria desde un medio de almacenamiento externo. Otras arquitecturas Harvard modificadas se comportan como Harvard puras en este ámbito.

Las memorias de datos e instrucciones pueden ser accedidas de diferentes formas Las máquinas Harvard puras, al tener distintos espacios de memoria para datos e instrucciones, pueden acceder a dichas memorias de distintas maneras. Puede ser el caso de una arquitectura donde la memoria de instrucciones se direccione con una cantidad de bits distinta a la de datos. Esto no es posible en una arquitectura von Neumann ni en una Harvard modificada de cache separado, dado que ambos espacios de memoria deben estar superpuestos.

Aplicaciones de esta clasificación en la actualidad

En áreas específicas donde un memoria cache es prohibitiva (puede ser el caso de los DSP por el determinismo necesario para las diversas operaciones, o de microcontroladores por el bajo costo), pueden darse arquitecturas Harvard puras o von Neumann. En modelos donde el uso es de propósito general, típicamente se utilizan arquitecturas Harvard modificadas con caché separado debido a que tener espacios de direccionamiento separados genera dificultados con ciertos lenguajes de programación de alto nivel que no soportan la noción de tener datos de sólo lectura cargados en un espacio de direccionamiento distinto al de los datos comunes, con la necesidad de utilizar distintas instrucciones para leerlos. El lenguaje de programación “C” soporta este comportamiento a través e extensiones propietarias, aunque hoy en día ya existen estandarizaciones para los llamados procesares embebidos.

2.2.2. Clasificación de las arquitecturas según su conjunto de instrucciones

RISC vs CISC

La ISA define el modelo lógico: Tres cosas, los registros, el modelo de la memoria y como se relacionan los registros (operaciones).

2.2.3. Taxonomía de Flynn

Instruction and data streams

IMPORTANTE:

El modelo formal del procesador queda definida por el ISA. El isa no es una tabla de instrucciones. Es el modelo lógico formal que describe la arquitectura.

Hablar de Alan Turing. Problemas Turing Computables -¿Investigar. Teorema de incompletitud de Gödel.

Referencias

[1] John von Neumann, “First Draft of a Report on the EDVAC”, 1945.

Luciano César Natale