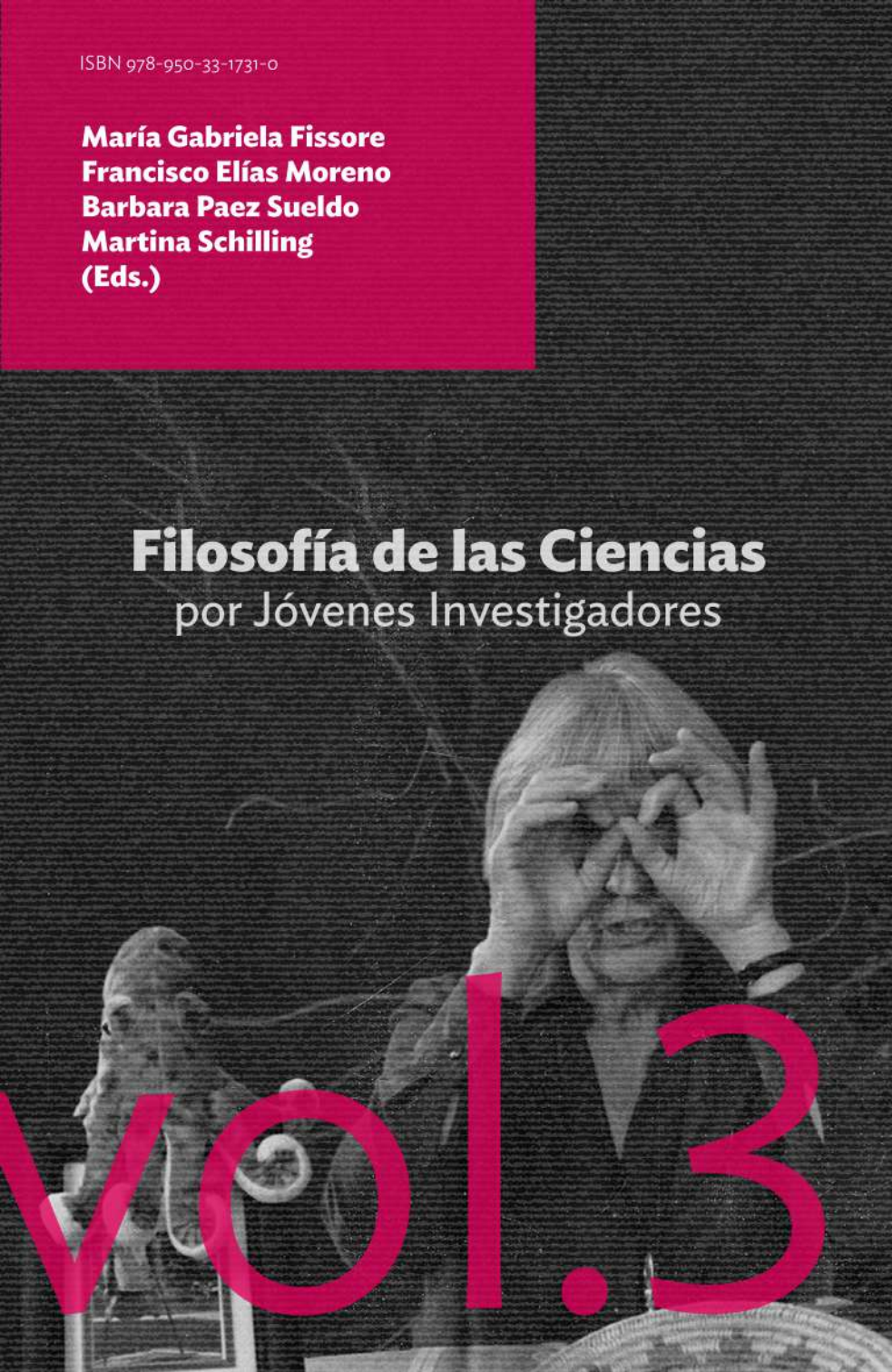
ISBN 978-950-33-1731-0

María Gabriela Fissore
Francisco Elías Moreno
Barbara Paez Sueldo
Martina Schilling
(Eds.)

Filosofía de las Ciencias
por Jóvenes Investigadores



Vol. 3

Filosofía de la Ciencia por Jóvenes Investigadores

vol. 3

María Gabriela Fissore
Francisco Elías Moreno
Barbara Paez Sueldo
Martina Schilling
(Eds.)

Colecciones del CIFFyH

Filosofía de la ciencia por jóvenes investigadores / Julián Arriaga... [et al.]; editado por Fissore María Gabriela... [et al.]. - 1a ed - Córdoba: Universidad Nacional de Córdoba. Facultad de Filosofía y Humanidades, 2023.
Libro digital, PDF

Archivo Digital: descarga y online
ISBN 978-950-33-1731-0

1. Filosofía de la Ciencia. I. Arriaga, Julián II. María Gabriela, Fissore, ed.
CDD 501

Publicado por
Área de Publicaciones de la Facultad de Filosofía y Humanidades - UNC
Córdoba - Argentina
1º Edición

Área de
Publicaciones

Diseño de portadas: Manuel Coll y Marìa Bella
Diagramación: María Bella

2023

Distinguir un mecanismo programable de uno proto-programable

Xavier Huvelle*

Introducción

El concepto de "programa" es visto en este trabajo desde la resolución de problemas como un plan o proceso para alcanzar un resultado. Se define "programa" de diversas maneras; como un plan semejante a los que realizamos en la mente (von Neumann & Morgenstern, 1944; Turing, 1948/1969), como una tabla de instrucciones o algoritmo (Dijkstra, 1988; Turing, 1947/2004) o como especificaciones (Duncan, 2017; Turner, 2011, 2018). Aquello que ejecuta el programa puede ser tanto una mente como una computadora, pero lo que nos interesa aquí no es realmente debatir el computacionalismo, sino lo que entendemos por "programa". Este trabajo trata de diferenciar mecanismos programables de mecanismos proto-programables, no desde una noción de computación, sino desde una de programación.

Por mecanismo entendemos un elemento compuesto de partes relacionadas entre ellas y que van formando un todo. De las relaciones de las partes y del todo se van produciendo distintos tipos de funciones y de las que podemos en principio determinar sus causas. En este trabajo usamos una noción de "mecanismo" que se asocia con la noción de máquina y la de sistema como solucionador de problemas. Esto es, una máquina o computadora es un mecanismo, aunque no cualquier mecanismo sino un mecanismo programable que busca alcanzar un resultado, una solución a partir de un conjunto de instrucciones. La noción de sistema es relevante para esta forma de caracterizar al mecanismo al igual que el concepto de organización. Esto se debe a que son características que limitan y restringen no solo las funciones que pueda realizar el artefacto, sino también las funciones que no puede hacer.

* CIFFyH (UNC).
Mail de contacto: xavier.huvelle@ffyh.unc.edu.ar

Un mecanismo programable visto en estos términos refleja por lo menos dos de las ideas principales de una máquina universal de Turing: 1. la máquina debe ser capaz de interpretar datos e instrucciones mediante un lenguaje codificado y 2., a partir de un cierto conjunto de datos y de instrucciones, podemos obtener múltiples funciones o resultados. Esto es, el mecanismo puede alcanzar una cierta generalidad en sus propósitos, pero también actuar de forma distinta a lo esperado. El punto 2. es una tesis computacional que se vincula con las nociones de computación y de lo computable. Un programa es entendido desde la perspectiva de la tesis de Church-Turing en la que todo algoritmo es una máquina de Turing y un programa es uno de estos algoritmos. Además, el punto 1. estipula la necesidad de un lenguaje de programación o codificado. A continuación, presentamos algunos casos muy antiguos de posibles programas que vemos como mecanismos proto-programables. En la sección que sigue, mostraremos una característica que nos permite distinguir mecanismos programables de proto-programables: la unificación de las instrucciones con los datos expresados en la arquitectura de von Neumann por permitir al programa ser almacenado físicamente en el mecanismo (y no distinguir datos de instrucciones). Otra ventaja de la arquitectura de von Neumann es que da lugar a una noción más clara y potente de sistema. A continuación, intentaremos responder a la pregunta: ¿qué hace que un mecanismo sea programable?

Casos

Una primera respuesta a dicha pregunta es que no todos los mecanismos proto-programables contienen programas, por lo que no pueden ser entendidos como computadoras ya que, por un lado: 1. no poseen un lenguaje de programación que puede ser usado para establecer instrucciones a otros mecanismos, y 2. su estructura física limita la capacidad del mecanismo para ejecutar más funciones que lo diseñado originalmente. No es, en este sentido, de propósito general. El concepto de "máquina universal" de Turing logra contener los principios de lo computable y no computable, así como los puntos 1. y 2., pero resulta muy general para ser visto como un modelo adecuado para la programación (De Mol, 2021; McCarthy, 1963). Una alternativa es ampliar la noción de Turing usando la idea de sistema de producción de Post (1936) o la del cálculo-λ de Church

(1933). Ambas formulaciones permiten ampliar la noción de computación a la de programación, Church con la tesis Church-Turing redefine al algoritmo (programa como algoritmo) y, en paralelo, Post con su idea de la resolución de un problema de decisión en pasos finitos (denominado "Fórmula 1"). Sin embargo, estas alternativas permanecen muy atadas a la noción de computación usada por Turing (cálculo efectivo) y puede resultar confusa para describir ciertos mecanismos proto-programables como, por ejemplo, el mecanismo de Anticitera o el teatro de Herón de Alejandría. Estos casos pueden ser vistos como mecanismos que poseen un programa y por ende ser entendidos como casos muy primitivos de mecanismos programables.

Citaremos tres casos particulares que marcan la diferencia entre lo que entendemos por mecanismos proto-programables.

Mecanismo de Anticitera

En mayo de 1902, entre los restos de una antigua embarcación que naufragó alrededor del siglo II a.C. en la costa de la isla de Anticitera en Grecia, Valerios Stais descubrió un extraño mecanismo. En lo que se había convertido en roca, se podía ver claramente un engranaje, dejando pensar que se trataba de un aparato que probablemente era un reloj astronómico. Es solo a partir de 1974 —tras la publicación de Derek John de Solla Price (1974) y Charalampos Karakalos— quienes usaron rayos X para estudiar los fragmentos, que se pudo confirmar la hipótesis de Stais. Este mecanismo, que es hoy conocido como "Mecanismo de Anticitera" (Figura 1), no solo permitía predecir posiciones astronómicas del cielo, sino también de eclipses. Algunos afirman que solamente recién en el siglo XIV, con el desarrollo de relojes mecánicos astronómicos en Europa, es que la complejidad poseída por el Mecanismo de Anticitera se puede comparar (Marchant, 2006). Una complejidad que otros como Efstathiou y Efstathiou (2018) consideran como el ejemplo más antiguo de computadora analógica. El mecanismo es capaz de predecir con mucha precisión los movimientos del Sol, la Luna y sus fases, los eclipses, los ciclos metóni-

cos[1], los saros[2], los exeligmos[3], ciclos calípicos[4] y los juegos olímpicos[5]. Otros piensan que podían también calcular los movimientos de los planetas conocidos de la época (Mercurio, Venus, Marte, Júpiter y Saturno) (Freeth et al., 2021). La introducción de los datos se realizaba a partir de una manivela o manija que hacía girar engranajes que movían las agujas sobre los discos de los paneles. El programa, en este caso, podría ser visto como la disposición de los engranajes unos con otros. Su particularidad es que para obtener nuevas funciones o cómputo se requeriría cambiar una buena parte de los engranajes (si no todo).

Figura 1: Modelo computacional de la máquina de Anticitera

Nota: Imagen tomada de "A Model of the Cosmos in the ancient Greek Antikythera Mechanism" por Freeth et al., 2021, *Scientific Reports, 11.*

1 Ciclo creado a partir de un común múltiplo aproximativo de los periodos orbitales de la tierra y de la luna.

2 Periodo de tiempo de 223 lunas o 6585.32 días en el que la luna y la tierra vuelven a sus órbitas y pueden reproducir eclipses.

3 Periodo de 55 años y 33 días para predecir eclipses con características similares a eclipses pasadas que usan este periodo.

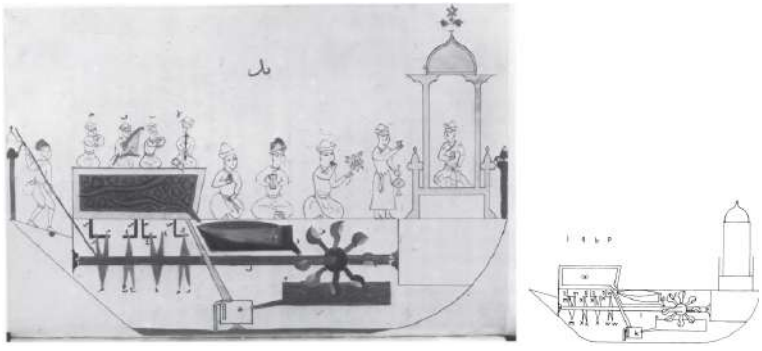
4 Ciclos de 76 años semejantes a los metónicos que se calculan a partir de un múltiple común del año y de los meses. La computación realizada por el mecanismo de Anticitera era muy precisa de 27758 días, el cálculo hoy es de 27758.8 días.

5 Que contenía una gran parte de los juegos panhelénicos y juegos menores.

Músicos de al-Jazari

En el siglo XII, un famoso inventor árabe de Jazira, en el norte de la actual Siria, llamado Ismail al-Jazari fue tan revolucionario que es considerado por muchos como el padre de la robótica por su gran manejo y uso de mecanismos. Entre una de sus tantas invenciones podemos destacar la de un autómata musical (Figura 2), que consistía en una pequeña embarcación con cuatro músicos autómatas que tocaban música. La embarcación contaba con un elaborado sistema en el que un disco con tachos, muy similar a los discos que animan las cajas musicales, interactuaba con paletas que producían el movimiento de los músicos y el de los instrumentos. A su vez, el disco giraba gracias a una pequeña rueda de agua alimentada por un reservorio que se rellenaba automáticamente cada 30 minutos permitiendo así la reproducción de una música durante unos 5 minutos. Se podía reordenar las paletas para recrear diferentes ritmos o canciones, haciendo de ese mecanismo un buen ejemplo de algo parecido a un mecanismo programable según Sharkey (2007).

Figura 2: Representación de los "esclavos" proviniendo del libro de Al Jazari

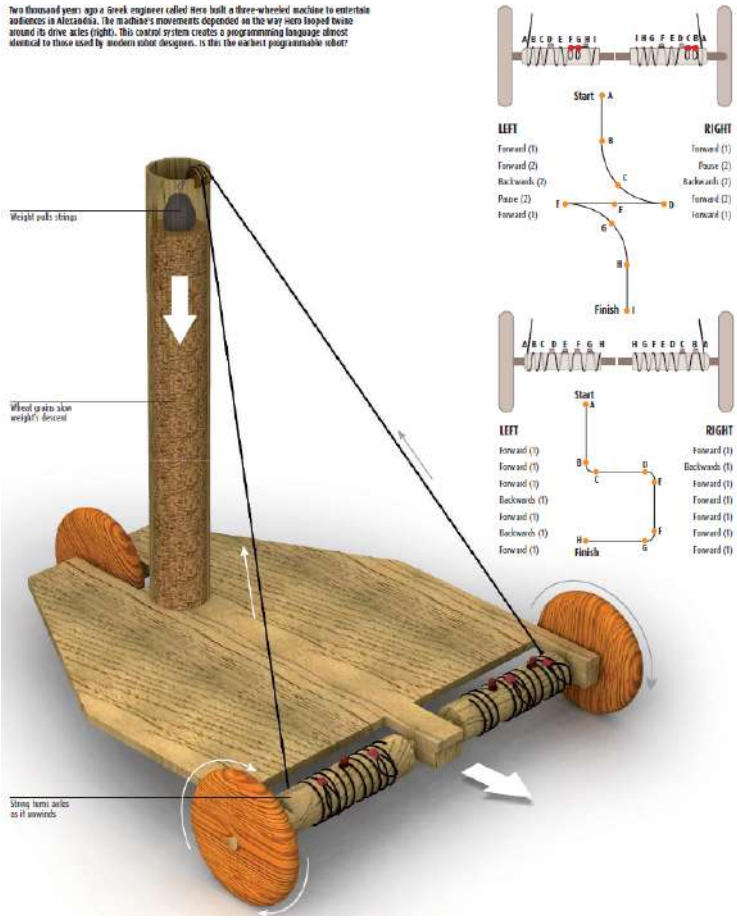


Nota: Imagen tomada de *The Book of Knowledge of Ingenious Mechanical Devices*, por Hill, 1974, p. 107.

Teatro de Herón

Sin embargo, existe otro ejemplo más antiguo todavía, según Sharkey, de un mecanismo programable que desafía aún más nuestras creencias sobre la antigüedad. La figura más emblemática es la de Herón de Alejandría con su teatro mecánico funcionando con cuerdas. El principio es relativamente sencillo (Figura 3): el escenario cuenta con tres o cuatro figurillas que se desplazan hacia adelante o atrás impulsado por un peso en caída ubicado en un tubo lleno de cereales que funciona como motor. El tubo posee un pequeño orificio en su base por lo que se puede medir el tiempo de caída del peso con precisión. Por otro lado, el peso se encuentra atado a dos cuerdas enrolladas en un eje horizontal que permite el movimiento de las ruedas hacia adelante pero además puede crear giros hacia la derecha o la izquierda. Lo impresionante de tal mecanismo es que la cuerda enrollada en el eje puede ser configurada de una cierta manera para permitir la creación de un patrón de movimiento bien específico. La configuración de la cuerda es realizada de una manera muy similar a las empleadas en programación a la hora de establecer las acciones deseadas para cada una de las ruedas. En el teatro de Herón de Alejandría el comando es central y depende entonces de la cantidad de veces que la cuerda se enrolle en el eje o de su interacción con las pequeñas paletas y cera (protuberancias en la Figura 3) para crear un patrón. Por ejemplo, si deseamos un movimiento hacia adelante y un retroceso, la cuerda es enrollada diez veces en diagonal desde la izquierda hacia la derecha y para retroceder se la cruza enrollada cinco veces desde la derecha hacia la izquierda. De este modo tenemos adelante (10) retroceso (5) conformando un tipo de lenguaje de programación y la posibilidad de programar o reconfigurar rápidamente movimientos muy específicos y deseados.

Figura 3: Prototipo de Sharkey mostrando el mecanismo usado por Herón de Alejandría para su teatro



Nota: Imagen tomada de "I ropebot" por Sharkey, 2007, *New Scientist, 195*(2611) p. 33.

Estos tres casos resultan interesantes porque muestran una cierta evolución respecto de los tipos de programas que poseen cada uno de estos mecanismos. La máquina de Anticitera no resulta muy flexible respecto

de la posibilidad de computar nuevas funciones. Su programa se encuentra totalmente vinculado con la estructura mecánica que la compone y la precisión exigida por sus partes es muy alta. Un mínimo defecto en sus engranajes puede resultar en imprecisión en el resultado (lo que podría haber ocurrido por la durabilidad[6] de los materiales). Por otro lado, el mecanismo requeriría modificaciones estructurales muy importantes para cambiar sus funciones y, por lo tanto, para poder introducir en el mecanismo nuevos conjuntos de instrucciones. En el mecanismo de al-Jazari existe una posibilidad de cambiar las instrucciones modificando el sistema de paletas: al modificar las paletas se puede cambiar el ritmo y la melodía de la canción producida por el mecanismo. En este caso —y a diferencia del mecanismo de Anticitera— se puede cambiar un patrón dentro del mecanismo para producir un nuevo resultado. Los cambios físicos y estructurales son menores, sin embargo, sus funciones siguen siendo limitadas. Tampoco existe en el mecanismo de al-Jazari algún tipo de código o lenguaje que permita dibujar nuevos patrones en el mecanismo. El teatro de Herón resulta muy interesante comparado con los dos tipos de mecanismos anteriores, ya que este mecanismo fue uno de los primeros en poseer un código que permitiera transcribir funciones particulares del mecanismo fuera de su estructura física. El propio Herón codificó patrones con el fin de favorecer su reproducción en futuras ocasiones. El motivo principal que lo condujo a ello fue por los problemas que padeció al intentar recrear los mecanismos de teatros automatizados más antiguos, por no disponer de ese tipo de codificaciones. Este teatro podría ser considerado como realmente un primer mecanismo programable, dónde primero se codifica el comportamiento deseado para luego materializarlo en los ejes y así recrear los movimientos deseados. Pero si el teatro de Herón posee un lenguaje de programación o proceso de codificación y tiene una gran flexibilidad en los tipos de patrones que puede crear, ¿por qué no es considerado un programa?

El teatro de Herón no puede ser considerado como un programa por dos razones: 1. los errores —cuando pueden ocurrir— son más problemáticos de corregir, se debe corregirlos físicamente dentro de la estructura misma del mecanismo y 2. posee una separación física entre los datos y las instrucciones. Si existiera un error o un desperfecto en el eje, inde-

6 Los engranajes podían ser dañados por el uso, ya que la calidad de los materiales no estaba a salvo de algún tipo de desgaste.

pendientemente de sus consecuencias (desestabilización del mecanismo, comportamiento no deseado, etc.), corregirlo requeriría no solamente eliminar el desperfecto sino volver a ubicar adecuadamente la cuerda en los pasos anteriores al desperfecto. Por otro lado, existe también una separación entre los datos y las instrucciones a nivel físico. En efecto, una de las características importantes de los programas es poder reusar datos producidos como instrucciones, pero para poder hacerlo se requiere que se encuentren almacenados en un mismo lugar físico. Esto se debe a que es necesario que el dato producido sea almacenado en un mismo espacio, para luego ser tomado —si se requiere— como una instrucción para generar otro proceso. En el teatro de Herón, las instrucciones y los datos se encuentran presentes en el eje, pero en este caso, el mecanismo es incapaz de poder distinguirlos. La distinción se da cuando se compara la codificación externa con la implementación (organización de los hilos y la cera) en el eje del mecanismo por un intérprete (el codificador, la persona que escribió el código). Este intérprete es externo al mecanismo. Esta última característica de la unificación entre datos e instrucciones no está presente en ninguno de los tipos de mecanismos citados anteriormente.

¿Qué hace único un programa computacional?

Algunas nociones como las funciones recursivas (Gödel, 1934/1986; Kleene, 1952) y la reflexividad (Blanco, 2017; Cantwell Smith, 1982) ofrecen herramientas conceptuales más adecuadas para poder establecer distinciones claras entre un mecanismo proto-programable y uno programable. La reflexividad, en particular, es la capacidad de poder acceder al propio código e implementar cambios para luego ejecutarlos. Es una característica única que disponen los programas computacionales frente a otro tipo de mecanismos. Una de las grandes ventajas de ambas nociones es que pueden generar nuevas funciones y nuevos comportamientos a partir de la ejecución del código y la interacción del programa con el entorno o el usuario. Cambios que crean una novedad dentro de las iteraciones generadas por el programa, que a su vez crean nueva información. Una de las dificultades de las nociones de recursividad y reflexividad es que pudieron ser implementadas en un lenguaje de programación recientemente (en 1959 y 1960) en los lenguajes LISP y ALGOL60, lo que descartaría —en cierta medida— cualquier noción previa de programa.

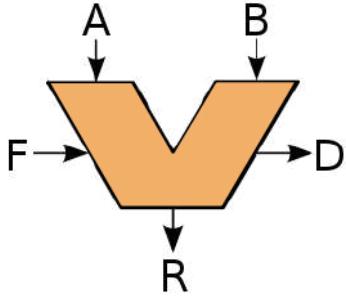
Una propuesta para ampliar y delimitar lo que podemos entender por "programa" es su característica de almacenar información e instrucciones, no de forma separada como la máquina de Babbage o del telar de Jacquard, sino internamente: como lo propuso teóricamente Turing y como se completó a través de la arquitectura de von Neumann por Eckert y Mauchly. Esta característica permite distinguir la *Mark I* de la EDVAC dónde la primera computadora (Mark I) separaba físicamente las instrucciones (en cintas perforadas) de los datos (interruptores electromagnéticos) por la denominada arquitectura de Harvard, mientras que la arquitectura de von Neumann unificaba el almacenamiento de datos e instrucciones en un mismo lugar físico (RAM). Esta es la primera característica que separa un mecanismo proto-programable de uno programable, donde los datos y las instrucciones se encuentran almacenados en un mismo espacio físico dentro del mecanismo y al que el sistema pueda acceder por sí mismo. Esta característica es extensiva con la recursividad, donde tantos los datos como las instrucciones pueden ser modificados para cumplir otros propósitos distintos a los originales. Además de la recursividad, existe otra característica que se ve fortalecida por la posibilidad de almacenar datos e instrucciones internamente: la de considerar al mecanismo como un sistema que requiere una organización y una jerarquía en la ejecución de sus procesos. Esta noción de sistema ya se encuentra presente en muchos de los mecanismos proto-programables, pero es aplicado en un mecanismo programable que permite el almacenamiento de datos e instrucciones en un mismo espacio físico es una característica única que sólo los programas poseen: pueden ejecutar distintos sistemas en paralelo con los mismos datos y un cierto conjunto de instrucciones. Esto es, existen niveles de abstracciones que permiten la coexistencia de distintos sistemas y subsistemas que se ordenan en diferentes jerarquías dentro del mecanismo. Uno de estos, propuesto por von Neumann, es la ALU[7] (*Arithmetic Logic Unit*) o unidad aritmética lógica, que permite realizar operaciones aritméticas y operaciones lógicas que no solamente organiza y estructura al sistema, sino que posee niveles de abstracciones distintos. Estos niveles permiten interpretar datos de entradas de formas distintas —tanto si fuera un valor binario (0 y 1) o un valor lógico (V o F)— permitiendo ser interpretados por distintos sistemas organizados jerárquicos que tomarán algunos de

7 La ALU podría ser contemplada como la aplicación de la diagonalización gödeliana por parte de von Neumann en computación.

estos dos valores como datos de entradas. El mismo valor binario puede ser interpretado mediante el uso de tablas como ASCII, UTF-8 o UNICODE que asocian valores binarios con un valor aritmético y con símbolos (letras, números, signos, etc.).

Existen varios tipos de ALU con capacidades distintas y también puede haber más de una en una CPU. La forma tradicional, sin embargo, es la de la Figura 4.

Figura 4: Representación gráfica de una Unidad Lógica y Aritmética o ALU.



Nota: Imagen tomada de Unidad aritmética lógica (2022, junio 23) en *Wikipedia*. https://es.wikipedia.org/wiki/Unidad_aritm%C3%A9tica_l%C3%B3gica#/media/Archivo:ALU_symbol.svg

Entendiendo de forma simplificada y rudimentaria el funcionamiento de la ALU, se puede observar que una de sus grandes funciones es la de convertir unos valores en otros. Allí actúan operadores que poseen un grado mayor de complejidad a la observada en una organización simple de puertas lógicas. Según el sistema jerárquico organizado, la información que aparece en el input puede ser "00110010", "ffvvffvf", o "2" según el sistema que lo interpreta. La ALU es un programa dónde el resultado R vuelve a ser almacenado en los registros para continuar las operaciones o directamente en la RAM si la unidad de control lo establece así. Tenemos aquí un primer programa que puede operar automáticamente con datos e

instrucciones almacenados en sus componentes, pero también modificar e interactuar con ellos[8].

Referencias bibliográficas

Blanco, J. (2017). Reflexiones sobre la reflexión. En A. Crelier y N. Fernández (Eds.), *La diferencia antropológica: Humano, animal, cyborg* (pp. 37-49). Mar del Plata: Universidad Nacional de Mar del Plata.

Cantwell Smith, B. (1982). *Procedural Reflection in Programming Languages* (Tesis de doctorado no publicada). Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge.

Church, A. (1933). A Set of Postulates for the Foundation of Logic (Second Paper). *Annals of Mathematics, 34*(4), 839-864.

De Mol, L. (2021). Turing machines. En E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Winter 2021). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/win2021/entries/turing-machine/

De Solla Price, D. J. (1974). Gears from the Greeks: the Antikythera mechanism, a calendar computer from ca. 80 B.C. *Transactions of the American Philosophical Society (New Series), 64*(7), 1-70.

Dijkstra, E. W. (1988). *A Method of Programming*. Boston: Addison-Wesley.

Duncan, W. (2017). Ontological Distinctions between Hardware and Software. *Applied Ontology, 12*(1), 5-32.

Efstathiou, K., y Efstathiou, M. (2018). Celestial Gearbox. *Mechanical Engineering, 140*(09), 31-35.

8 Deseo agradecer a la organización del evento y especialmente al comentador del trabajo quién ofreció mucha información y materiales para mejorar el trabajo pero que, por falta de espacio, no se pudo insertar en el presente escrito.

Freeth, T., Higgon, D., Dacanalis, A., MacDonald, L., Georgakopoulou, M., y Wojcik, A. (2021). A Model of the Cosmos in the ancient Greek Antikythera Mechanism. *Scientific Reports, 11*(1), 5821.

Gödel, K. (1986). On undecidable propositions of formal mathematical systems. En S. Feferman, John W. Dawson, Jr., Stephen C. Kleene, G. Moore, R. Solovay, y Jean van Heijenoort (Eds.), *Kurt Gödel: Collected Works: volume I: publications 1929-1936* (pp. 338–371). Oxford: Oxford University Press. (Trabajo original publicado en 1934)

Hill, D. R. (1974). *The Book of Knowledge of Ingenious Mechanical Devices.* Dordrecht: D. Reidel Publishing Company.

Kleene, S. C. (1952). *Introduction to Metamathematics.* Amsterdam: North-Holland Publishing.

Marchant, J. (2006). In search of lost time. *Nature*, *444*, 534-538.

McCarthy, J. (1963). A Basis for a Mathematical Theory of Computation. En P, Braffort y D. Hirschberg (Eds), *Computer Programming and Formal Systems* (pp. 33-70). Amsterdam: North-Holland.

Post, E. L. (1936). Finite Combinatory Processes-Formulation 1. *Journal of Symbolic Logic, 1*(3), 103-105.

Priestley, M. (2011). *A Science of Operations.* Berlin: Springer-Verlag.

Sharkey, N. (2007). I ropebot. *New Scientist, 195*(2611), 32-35.

Turing, A. M. (2004). Lecture on the Automatic Computing Engine. En J. B. Copeland (Ed.), *The essential Turing* (pp. 378-394). Oxford: Oxford University Press. (Trabajo original publicado en 1947)

Turing, A. M. (1969). Intelligent Machinery. En B. Meltzer y D. Michie (Eds.), *Machine Intelligence 5* (pp. 3-26). Edinburgh University Press. (Trabajo original publicado en 1948)

Turner, R. (2011). Specification. *Minds and Machines*, 21, 135-152.

Turner, R. (2018). *Computational Artifacts: Towards a Philosophy of Computer Science*. Berlin: Springer Berlin Heidelberg.

von Neumann, J., y Morgenstern, O. (1944). *Theory of games and economic behavior*. Princeton: Princeton University Press.