Creación de un ecosistema donde preservar el primer lenguaje y compilador argentino: Un caso de arqueología computacional.

Gustavo del Dago
Proyecto SAMCA; gdeldago@gmail.com

***Resumen*—El lenguaje y compilador *COMIC* fue desarrollado en el Instituto de Cálculo (*IC*) de la Facultad de Ciencias Exactas y Naturales de la Universidad de Buenos Aires (FCEyN UBA) durante los años 1965 y 1966. El contexto donde se llevó a cabo esta experiencia, que hoy es considerada como la primera de su clase en Argentina, merece especial atención: las condiciones de aislamiento en que se encontraba el *IC* durante estos años, las técnicas de ingeniería inversa empleadas en el diseño y desarrollo, las dificultades planteadas por la escasez de documentación y el final abrupto de las actividades; son cuestiones fundamentales a la hora de analizar retrospectivamente los resultados. Consideramos esencial para el estudio de las técnicas y herramientas con las que se trabajó en el *IC* tener la posibilidad de analizar los productos obtenidos. El producto de mayor relevancia en el marco de este proyecto de investigación es el *COMIC*, un programa de computación especialmente desarrollado para una máquina de cual ya no se dispone de ningún ejemplar. Este trabajo tiene como principal objetivo exponer el proyecto de preservación del *COMIC* para el que se empleó una metodología que tiene como eje central la construcción de un *ecosistema* donde se puedan ejecutar antiguos programas de computación sin necesidad de contar con los equipos originales. Se presentarán las etapas seguidas durante el proyecto, se dará cuenta del estado de evolución del mismo, exponiendo los primeros resultados y mostrando como el *ecosistema* creado se convierte al mismo tiempo en un elemento habilitante para futuros estudios. Estudios que pueden trascender a los específicamente técnicos alcanzando un *ecosistema ampliado* que incluya al resto de las actividades y personas que conformaron el *IC*.**

***Abstract*—The computer language and compiler *COMIC* was developed at the Instituto de Cálculo (*IC*), Facultad de Ciencias Exactas y Naturales de la Universidad de Buenos Aires (UBA FCEyN) during the years 1965 and 1966. The context where this experience took place, which today is considered the first of its kind in Argentina, deserves special attention: the isolation of the *IC* at that time, the reverse engineer techniques employed in the design and development, the difficulties come from the lack of documentation and the sudden end of the activities, are key issues when analyzing retrospectively the results. We foresee essential for the study of the techniques and tools implied at the *CI* to be able to analyse the resulting products. The most important product in the context of this research project is the *COMIC* itself, a computer program specially developed for a machine which is no longer available. The main objective of this paper is to introduce the project on itself to preserve *COMIC*, where we used a methodology that prioritizes an *ecosystem* creation in order to run old computer programs The actual work will introduce the different project stages, expose the first results and show how the proposed *ecosystem* becomes a space where future studies can take place, maybe transcending the specifically technical issues and reaching the *extended ecosystem* that includes the rest of the activities and people who formed the *IC*.**

***Index Terms*—Ferranti Mercury, Clementina, COMIC, Instituto de Cálculo, Assembler PIG, Input Routine, Autocode, Software preservation**

I. Introducción

A. El pasado de la computación. Ciclo de vida de los sistemas

La tecnología mantiene un avance continuo, el resultado de este avance nos permite contar en muchos casos, con equipos de computación más potentes en cuanto a velocidad de proceso, capacidad de almacenamiento y facilidades de utilización. En este escenario los sistemas de computación se ven expuestos a una rápida obsolescencia. Un programa de computación puede describirse como la representación de un algoritmo según reglas de codificación impuestas por las características de una máquina específica o sistema objeto. Los programas se pueden hallar tanto en código fuente como código objeto o binario [1]. En consecuencia los programas de computación dependen de la plataforma o equipo para los que fueron creados, y en principio su existencia está supeditada a la dichos equipos o plataformas operativas. Es probable que la utilidad de algunos programas se extienda en el tiempo sobreviviendo a los equipos utilizados para su ejecución, situación que motiva la creación de varios métodos para facilitar la migración de programas y datos entre distintas plataformas. Herramientas que transcriben código en forma automática, convierten datos o emulan distintas arquitecturas donde ejecutar antiguos programas, son algunas de las soluciones que ha ofrecido la industria de la computación a fin de posibilitar la actualización o reemplazo de equipamiento sin perder las inversiones realizadas en el desarrollo de los programas[2]. La oferta de soluciones expuesta está acotada en el tiempo. Comienza en un momento donde la inversión en el desarrollo de programas es considerable y finaliza generalmente cuando el esquema ya no resulta conveniente desde el punto de vista económico.

Finalmente llega la instancia en que los programas se consideran obsoletos y se dejan de utilizar. De no tomarse los recaudos para su preservación, una parte del pasado de la computación se pierde de manera irrecuperable.

[1] El código objeto o binario normalmente es generado en forma automática a partir de un proceso de compilación o ensamblado.

[2] La compañía *IBM* constituye un ejemplo del sector comercial pues se caracterizó por ofrecer a sus clientes gran cantidad de sistemas con capacidades de emular modelos anteriores a fines de facilitar el cambio de equipamiento.

Paradójicamente el progreso de la tecnología aplicada a la computación contribuye a eliminar algunos vestigios de su propia historia.

B. Arqueología computacional. Técnicas y Herramientas

Las disciplinas relacionadas con la preservación del legado computacional se encuentran actualmente en un estado embrionario. En *Introducción a la Arqueología Informática* [1] se propone un método de trabajo y se describen algunas herramientas tendientes a facilitar las distintas tareas que, en conjunto, permiten disponer de los objetos preservados. Estos objetos se convierten, según nuestra visión, en la materia prima con la que podremos abordar estudios en profundidad sobre los entornos a los que pertenecieron. Los programas de computación, son, sin duda una parte importante del patrimonio cultural heredado. El estudio de un programa de computación, se puede encarar de una forma estática, mediante la lectura de su código fuente, o dinámica, observando los resultados de su ejecución en un sistema de cómputo. Como veremos, la conservación de los objetos físicos asociados presenta una serie de dificultades y en consecuencia es muy frecuente que se disponga de programas para los cuales ya no existen equipos capaces de ejecutarlos. En el presente trabajo se propone como metodología principal la creación de un *ecosistema* donde la ejecución de esos programas sea posible. Por lo tanto la adopción de algunas prácticas y el uso de las herramientas descriptas en [1], con algunas variaciones, tendrá sentido sólo en el marco de la construcción de un *ecosistema* determinado.

Según la Real Academia Española, se define *arqueología* como "Ciencia que estudia lo que se refiere a las artes, a los monumentos y a los objetos de la antigüedad, especialmente a través de sus restos"[3]. Atendiendo a la definición anterior podemos convenir el uso de *arqueología computacional* para referirnos al conjunto de tareas y actividades relacionadas con la preservación y el estudio de los sistemas de computación antiguos, con el objeto de analizar y comprender su diseño y funcionamiento.

Como hemos visto, el ciclo de vida de los sistemas de computación representa un problema para quienes se interesen por estudiar en detalle sistemas obsoletos o extintos. Los equipos suelen pasar a desguace y en consecuencia algunas partes terminan convertidas en chatarra o reutilizadas en la construcción de nuevos dispositivos. Los programas y la documentación, tal vez debido a la facilidad con la que se pueden obtener copias, muchas veces corren mejor suerte y son conservados por personas o instituciones. En consecuencia las distintas partes componentes de los sistemas suelen estar fragmentadas, incompletas o con un alto grado de dispersión geográfica, situación que le confiere a la tarea del investigador de la historia de la computación su carácter arqueológico.

El proceso de preservación de programas se divide, a grandes rasgos, en las etapas descriptas a continuación.

1) Localización y recopilación de objetos materiales: Los objetos materiales pueden encontrarse en museos o instituciones[4] que realizan tareas de conservación de equipos de computación y tecnológicos. También es frecuente que profesionales y aficionados, en grupos y asociaciones o en forma independiente, guiados muchas veces por una cuestión nostálgica, lleven adelante tareas de coleccionismo, preservación, restauración y divulgación de sistemas antiguos.

El trabajo de recopilación de material consiste en localizar y obtener la mayor cantidad de objetos componentes de los sistemas que se pretende preservar. Comprender la importancia de cada una de las partes u objetos hallados y sus relaciones con el sistema que conformaron en el pasado es de vital importancia. Comprensión y conocimiento que posiblemente no tengan quienes conservan los objetos físicos. De manera que la localización y recopilación de material a la que aquí se hace referencia, incluye un análisis que permita establecer las relaciones de funcionalidad y, en algunos casos, de temporalidad entre los objetos disponibles.

A continuación se enumeran los objetos de mayor relevancia[5] y algunas de sus características incluyendo un resumen de los métodos y técnicas con que contamos para su recopilación y preservación.

a) Programas: Los programas de computación tienen existencia como objetos materiales sólo cuando se los almacena en un medio o soporte físico. Los medios de almacenamiento de datos y programas en tanto productos tecnológicos son objeto de constante evolución. Cintas y tarjetas de papel perforado, cintas y discos magnéticos o copias impresas son algunos de los soportes de información que se han utilizado en el pasado o se utilizan en la actualidad. Dependiendo del caso será necesario desarrollar herramientas que permitan trabajar con los contenidos disponibles. La conversión de formatos lógicos, la construcción o restauración de equipos periféricos o la transcripción manual de datos son algunas de las actividades necesarias para la preservación de programas.

b) Documentación: De acuerdo a la época a la que pertenezcan los documentos disponibles, puede ser que la información se encuentre en forma manuscrita, en mal estado de conservación o en copias de baja calidad. Estos factores justifican el trabajo de copia o transcripción a formatos digitales.

c) Testimonios: Los relatos de las personas involucradas en el desarrollo y uso de los sistemas tienen un valor fundamental a la hora de reconstruir o estudiar entornos de trabajo extintos. La documentación, cuando se dispone de ella, muchas veces es incompleta o contiene errores y los equipos físicos posiblemente ya no estén disponibles para su estudio. Los recuerdos de los protagonistas, aún con las limitaciones naturales en cuanto precisión o a nivel de detalle, pueden ser la clave para completar o enriquecer el material disponible. Recuperar y documentar dichos testimonios es un trabajo de índole iterativa que incluye etapas de obtención

[3]DICCIONARIO DE LA LENGUA ESPAÑOLA - Vigésima segunda edición

[4]Una importante cantidad de material relacionado con la máquina *Mercury* se encuentra en el MOSI (Museum of Science Industry). Uno de los más prestigiosos grupos de preservación de máquinas y programas de los primeros años de la computación electrónica lo constituye el Computer Conservation Society

[5]El grado de relevancia de cada objeto tendrá correspondencia con el tipo de proyecto. En este caso se trata de la preservación de un programa de computación

de testimonios, interpretación de los mismos, obtención de conclusiones y datos, análisis compartido de las conclusiones y obtención de nuevos testimonios que permitan validar y refinar las descripciones de los datos expuestos.

2) Emulación. Construcción de entornos virtuales: La razón de ser de un programa de computación es el momento de su ejecución y la posterior obtención de resultados. Cuando los equipos para los que un determinado programa fue diseñado ya no están disponibles se pierde la posibilidad de ejecutarlo. Las técnicas de emulación son la estratégica que permite crear un *ecosistema* donde los programas objeto de estudio puedan seguir funcionando. Este ecosistema estará conformado, no sólo por el equipo de cómputo y sus periféricos en versiones virtuales o emuladas, sino también por las Rutinas de Base, Sistemas Operativos y Bibliotecas de Funciones de las cuales los programas dependan.

Un emulador es un tipo particular de programa cuya función es replicar el comportamiento de distintos equipos y dispositivos físicos, a fines de ofrecer a los programas preparados para estos últimos un entorno de ejecución indistinguible del original. [6]

Disponer de entornos de trabajo virtuales o emulados tiene una importancia clave. Si bien es cierto que se pueden analizar programas con la simple inspección de su código fuente o incluso de su código binario, es la instancia de ejecución, que idealmente ofrece capacidades de depuración y ejecución controladas muchas veces no disponibles en los equipos originales, la que permite aumentar nuestra comprensión sobre estos objetos. Los resultados obtenidos de las ejecuciones de un programa, aún en el caso de ejecuciones parciales, suelen constituir nuevos elementos de estudio que permiten validar o aumentar la documentación, o confirmar hipótesis planteadas sobre su funcionamiento.

3) Adquisición de Experiencia: Para recrear un sistema de computación extinto, no alcanza con la construcción de un entorno de emulación, pues es indispensable contar con la experiencia y conocimientos que permitan su operación.

Las prácticas o modos de uso relacionadas con un sistema pueden incluir saberes no documentados, saberes que en la dinámica de trabajo pueden ser compartidos y aplicados. De modo que la falta de documentación no representa necesariamente un inconveniente. Sin embargo con el paso del tiempo o la falta de aplicación, este conocimiento se podría perder.

En los proyectos de preservación donde participan personas que han trabajado con los sistemas originales será muy valioso documentar prácticas o formas de uso. En cualquier caso, operar los sistemas virtuales, utilizar y verificar el funcionamiento de sus componentes o desarrollar nuevos programas son ejercicios que ayudarán a adquirir la experiencia necesaria.

[6]Un caso particular de emulación se presenta durante el desarrollo de nuevos equipos, donde los emuladores permiten contar con versiones virtuales de los equipos en etapa de producción. De esta forma el desarrollo de programas de base y utilidades se puede comenzar antes de que los equipos se terminen de construir. En el año 1963 Alicia Susana Chacur con la colaboración de Victoria Bajar y la dirección de Ernesto Garcia Camarero, desarrollaron un programa denominado *PICME* (*Programa Interpretativo de CEUNS por Mercury*) [2] que permitió desarrollar en la *Mercury* del *IC* los programas de base de la computadora *CEUNS* [3], máquina diseñada en la Universidad Nacional del Sur por el Ingeniero Jorge Santos y que se encontraba en proceso de construcción

C. Breve introducción al COMIC

El lenguaje y compilador *COMIC* (COMpilador del Instituto de Cálculo) es considerado el primer desarrollo de su clase realizado en Argentina. Esta característica sumada a la relevancia del instituto donde se gestó [7]y los hechos que ponen fin a su desarrollo [8], lo convierten en un objeto de especial interés técnico e histórico. El *COMIC* fue desarrollado en el Instituto de Cálculo (*IC*) de la Facultad de Ciencias Exactas y Naturales de la Universidad de Buenos Aires (FCEyN UBA) [9] entre los años 1965 y 1966 por la sección de Sistemas de Programación [10] liderada por Wilfred O. Durán [11].

El *COMIC* es un derivado del *AUTOCODE* que escribiera en el año 1957 R. A. Brooker para la máquina Mercury[12] de la compañía Ferranti [10] [11] [12]. El desarrollo del *COMIC* se plantea como una forma de superar las limitaciones impuestas por el *AUTOCODE* frente a los trabajos de programación que se realizaban en el *IC* que ya no se limitaban estrictamente al área matemática[13]. Una exposición detallada del *COMIC* que incluye descripciones de sus características y notas históricas sobre su desarrollo puede verse en [9].

Vamos a mencionar aquí y a modo ilustrativo, algunos de los cambios y mejoras incorporadas. Una modificación ensayada al evaluador de expresiones permitió utilizar identificadores de variables con longitud arbitraria [14]. En el lenguaje original esta longitud se limitaba a un caracter, decisión que según el mismo autor estaba en concordancia con el uso esperado para el lenguaje: la solución de problemas de índole matemática, disciplina donde se acostumbra nombrar a las variables con una sola letra o símbolo [14]. Esta decisión de diseño hizo que en la sintaxis del lenguaje, sea suficiente la simple yuxtaposición de factores para expresar el producto. En *COMIC* fue necesario incorporar un símbolo, se eligió el asterisco, para indicar la operación producto en forma explícita. También se incorporaron al lenguaje nuevos tipos de datos para gestionar variables booleanas y matriciales, facilidades para la operación con estos nuevos tipos de datos (ahora nativos) y una variedad de rutinas frecuentemente utilizadas en los programas. Especial atención merecen las rutinas para la generación de gráficos mediante la utilización de un dispositivo graficador y el mejor aprovechamiento de la memoria física a partir de una modificación realizada a la máquina en el mismo *IC*. Más

[7]Es importante destacar que el desarrollo de programas no constituía uno de los objetivos que dieron origen al *IC*, de hecho la sección Sistemas de Programación se crea con la intención de resolver en el ámbito local las dificultades o limitaciones de los sistemas que se estaban utilizando

[8]El desarrollo del *COMIC* se ve interrumpido con el golpe de estado y la intervención militar al Instituto de Cálculo. Una crónica de los hechos se puede consultar en [4] y en [5]

[9]Para conocer sobre el Instituto de Cálculo *IC* se puede consultar [6], [7] y [8]. Algunos ejemplares del Informativo del Instituto de cálculo están disponibles en el sitio WEB El Granero Común elgranerocomun.net

[10]La sección Sistemas de Programación se denominaba originalmente Sistemas, con la decisión de desarrollar un nuevo lenguaje de programación cambia su nombre a Sistemas de Programación.

[11]El equipo estable lo completaban Cristina Zoltan y Clarisa D. Cortés a quienes se sumaron Liana S. Lew y Noemí Garcia con dedicación parcial. [9]

[12]Esta es la máquina con que contaba el Instituto de Cálculo.

[13]Los tipos de trabajos que se realizaban se pueden consultar en los boletines del *IC*

[14]Los identificadores pueden tener entre 1 y 56 caracteres alfabéticos o los dígitos del intervalo [0-4], de los cuales los primeros 6 son significativos. [13]

adelante se volverá sobre algunos de estos aspectos.

El nuevo lenguaje y compilador está constituido por la suma de estas mejoras y agregados. Cambios que pueden parecer a simple vista triviales, pero que constituyeron un gran esfuerzo por parte del equipo de trabajo. Fundamentalmente porque no se disponía ni del código fuente ni de la documentación de diseño del *AUTOCODE*. Para salvar este problema se trabajó partiendo del código binario y utilizando técnicas que hoy conocemos como ingeniería inversa.

Incorporar el reconocimiento de expresiones más complejas que permitan manejar variables con nombres de longitud arbitraria, nuevos tipos de datos para álgebra booleana y matricial o el reconocimiento de paréntesis para indicar precedencias en los cálculos, entre otras modificaciones, requiere de cambios importantes en las rutinas o módulos de análisis léxico y semántico del compilador.

El abordaje del *COMIC* requiere considerarlo como un trabajo en curso y no como un producto terminado puesto que las actividades del *IC* fueron interrumpidas abruptamente impidiendo el desarrollo de todo su potencial.

II. Proyecto de preservación del *COMIC*

A continuación haremos un resumen de las distintas tareas llevadas a cabo en el proyecto de preservación del *COMIC* agrupadas en cada uno de los frentes: objetos materiales, emulador, programas de base y el propio lenguaje y compilador *COMIC*. Cuando se considere oportuno se justificará la necesidad de ciertas actividades, tratando de mostrar en todos los casos que el método propuesto ayudó a conseguir el resultado final. Se expondrán de forma muy sintética las cuestiones más salientes de cada apartado.

1) Objetos materiales: La idea de construir un emulador de la máquina *Mercury* se gestó de manera casi inmediata al comprobar que no se disponía de elementos que permitieran recrear el entorno de trabajo con que contaban los pioneros de la computación en la Argentina durante los años setenta. Descubrir que aún se conservaban copias de algunos programas y manuales publicados le dio una nueva dimensión a esta idea de proyecto que ya no tuvo como límite lograr la reconstrucción de una arquitectura computacional determinada sino que incluyó la preservación de estos programas y documentos. Wilfred Durán cedió a la FCEyN copias en cinta de papel perforado del *COMIC* y de su manual de programación. De esta manera se obtuvieron los primeros objetos materiales, que sin ser los únicos recolectados en este trabajo de investigación, se los puede considerar de una importancia clave, pues la posibilidad de contar con este material cambió el eje del proyecto hacia la recreación de un ecosistema que permita la ejecución de estos programas originales. Es muy probable que la recolección de objetos materiales tenga un nuevo impulso a partir de la difusión de los trabajos de preservación que se están llevando adelante.

2) Emulador: [15]

Existen en la actualidad gran cantidad de emuladores que permiten recrear distintos equipos y sistemas de computación. Los objetivos perseguidos por quienes crean emuladores pueden ser muy variados. Sin embargo, equipos y sistemas que han tenido gran difusión en el mercado, especialmente los diseñados para entretenimiento como sistemas de vídeo juegos o computadoras hogareñas de la era anterior al arribo de las computadoras personales actuales, representan el conjunto con mayor oferta. De hecho, es factible encontrar mas de un emulador para cada sistema. Los equipos utilizados en los inicios mismos de la computación, representan un caso especial. Un factor que atenta contra la disponibilidad de emuladores de estas máquinas es que no han sido equipos de uso masivo o personal. Por otra parte una gran cantidad de los programas desarrollados para estas máquinas se ha perdido, motivo que podría justificar un bajo interés en los propios sistemas. La falta de documentación o la dificultad en dar con la misma, las pocas unidades fabricadas de cada máquina [16] y la falta de experiencia en su utilización completan un panorama que explica la ausencia de emuladores de estos equipos o sistemas.

No se tiene registro de la existencia de algún emulador de la *Mercury*. El objetivo de preservación perseguido por este proyecto justifica a nuestro juicio, el desarrollo del primer emulador del Sistema *Mercury*. Para el trabajo de emulación se utilizaron como base documental, el manual del programador de la máquina del año 1960 [15] y el manual de programación en lenguaje convencional escrito por Ernesto García Camarero en el año 1962 [16]. Con la información disponible en estos manuales, muchas veces complementaria, se sintetizó en un documento la arquitectura de la máquina y sus periféricos. Detalles sobre los registros internos, organización de la memoria y métodos de acceso, tambores magnéticos o unidades de cinta, quedaron de esta forma definidas a partir de sus características funcionales. Es importante destacar que no se poseen datos sobre la estructura física de los dispositivos, lo que imposibilita su emulación a nivel de comportamiento eléctrico o electrónico. Aún con estas limitaciones, y como queda demostrado en el marco de este trabajo, una emulación funcional permite la ejecución de los programas binarios originales de la máquina en el entorno de emulación. El método que guió el diseño y programación del emulador no fue otro que el de inferir el funcionamiento de la máquina partiendo de las instrucciones o explicaciones sobre su programación. En líneas generales, podemos pensar que si determinadas instrucciones se utilizan para lograr algún efecto, el emulador deberá, frente a la ejecución de dichas instruccio-

[15] Al no haber consenso en el término utilizado muchas veces se utilizan como sinónimos emulador y simulador. En nuestra opinión el término "simulador" es más apropiado para definir el sistema que por el cual se simula algún aspecto de la realidad, tomemos por caso la simulación de un cambio climático y sus consecuencias. Por ese motivo el término "emulador" podría aplicarse con menos vaguedad a los procesos que recrean algún dispositivo del mundo físico, ofreciendo una versión virtual del mismo.

[16] De acuerdo a un resumen confeccionado en el año 2005 por el *Computer Conservation Society* sobre el documento *The Ferranti Computer Department – an informal history* B B Swann, 1975, la firma Ferranti fabricó solamente 19 equipos del modelo *Mercury*

nes, "emular" el efecto esperado. Supongamos el caso de una instrucción que realiza la operación aritmética suma tomando como operandos un registro y el contenido de una dirección de memoria. El resultado esperado es la actualización del registro, y seguramente de algún registro asociado como indicador de acarreo o cero, cuyo nuevo valor deberá ser la suma de los operandos intervinientes. Entendiendo la máquina o sistema emulado como un conjunto de objetos que se encuentran en un estado, la función del emulador no es otra que la gestión de las transiciones entre los distintos estados siempre de acuerdo a las reglas impuestas por la arquitectura de la máquina. Volviendo a nuestro ejemplo de la operación aritmética, el emulador implementa la transición de los estados para los objetos registro, indicador de acarreo e indicador de cero y gestiona asimismo el efecto de desbordamiento frente a operaciones cuyo resultado excede el tamaño de memoria de los distintos objetos. Se puede apreciar que aún sin conocer los mecanismos que permitieron construir la máquina original, ajustándonos a esta técnica de diseño, podemos desarrollar una máquina virtual que se comporte funcionalmente de la misma forma.[17] Información técnica sobre la arquitectura del emulador se puede encontrar en el sitio WEB del proyecto [17]

3) Programas de base: Input Routine[18] es el nombre genérico que se le dio originalmente al conjunto de programas que facilitan la operación básica de una computadora. Cuando aún no existían los Sistemas Operativos, los subsistemas de Entrada-Salida ni la estratificación por niveles en la arquitectura de los programas operativos, la *Input Routine* agrupaba una cantidad de utilidades de uso frecuente. En la máquina *Mercury*, la *Input Routine* incorpora, entre otras, facilidades para la carga y grabación de programas vía cinta de papel perforado, un programa ensamblador [19], los módulos de administración de Capítulos (Técnica previa al paginado de memoria) y una veintena de funciones de uso frecuente denominadas en el vocabulario de *Ferranti* como *Quickies* [20] Se puede trazar un paralelo entre la *Input Routine* por un lado y el conjunto constituido por los programas embebidos, sistema básico de entrada-salida y Sistema Operativo de los equipos actuales por el otro. A la hora de reconstruir un entorno de ejecución completo que simule la máquina *Mercury*, la *Input Routine* constituyen una pieza clave. Esta importancia se explica en virtud de la dependencia que tienen los programas desarrollados para la máquina respecto de los servicios y utilidades ofrecidos por la propia *Input Routine*. Es cierto que conociendo la funcionalidad ofrecida por estos servicios y utilidades se podría desarrollar un nivel de emulación funcional. Sin embargo contar con la *Input Routine* original permite por otra parte una recreación más realista del entorno y fundamentalmente constituye un elemento de prueba muy valioso a la hora de validar el funcionamiento del entorno de emulación de la máquina y sus periféricos. Volviendo a la cuestión del realismo en el entorno recreado, vamos a citar simplemente un ejemplo: una vez disponible el emulador de la máquina si no se cuenta con los programas de base, que incluyen la secuencia de arranque, estaríamos obligados a escribir algún tipo de asistente para la carga de programas de aplicación desde los dispositivos virtuales. En la *Mercury* la *Input Routine* residía de forma permanente en los tambores magnéticos. Los tambores de la máquina que existió en el *IC* no han llegado hasta nuestros días, de manera que no es posible conocer su contenido y por ende no se cuenta con una copia de la *Input Routine* que existió en aquel entorno. Los tambores magnéticos utilizados en los años sesenta del siglo XX raramente se han conservado enteros y no se conocen casos donde se haya podido rescatar la información almacenada. Se verá que es gracias a otra técnica de las décadas de 1950 y 1960 que hoy tenemos la posibilidad de reconstruir la *Input Routine* necesaria para nuestro proyecto. En aquellos años la codificación de programas se realizaba casi íntegramente en forma manuscrita, el código fuente original que contenía no solo las instrucciones sino también los comentarios y esquemas para el seguimiento del código, está manuscrito y en consecuencia el *medio* de almacenamiento es el papel. En el marco de este proyecto se localizó una copia original de dichos programas datada en Julio de 1957. Este ejemplar, probablemente único, está en manos de un coleccionista privado, profesor de la universidad de Edimburgo, que no solo facilitó los originales sino también los servicios prestados por la universidad para generar una copia en formato digital[21]. A partir de esta copia se pudo comenzar la tarea de transcripción manual a fines de obtener archivos digitales. Luego se desarrolló un ensamblador cruzado, que permitió la generación del código binario finalmente almacenado en los tambores magnéticos virtuales del emulador. El ensamblador cruzado se desarrolló en lenguaje C y está compilado para plataformas GNU/Linux y Windows. Se trata de un ensamblador simple de dos pasadas que permite la utilización de etiquetas para indicar direcciones de memoria o datos y directivas de ensamblado al estilo *PIG-2*. Sin bien la única salida que ofrece actualmente el ensamblador es un archivo binario de formato compatible con el emulador, una próxima versión incluirá la generación del listado ensamblador. El ensablandor cruzado es una herramienta clave para obtener de manera rápida y eficáz código binario compatible con la máquina emulada a partir de código fuente escrito para el sistema original.

4) COMIC*:* Como ya se anticipó en este mismo trabajo, el lenguaje y compilador *COMIC* fue desarrollado durante los años 1965 y 1966 y no se tiene constancia de hasta cuando se siguió utilizando o que cantidad exacta de programas de aplicación se desarrollaron con este lenguaje. Sin embargo podemos inferir de acuerdo a los testimonios de uno de sus autores [9] que la fecha en que dejó de utilizarse coincide con la de los sucesos conocidos luego como la noche de los

[17]La misma técnica de emulación es aplicable a sistemas con mayor cantidad de niveles de servicio, de forma que siempre es posible la emulación de un nivel inferior. Como ejemplo de lo anterior podemos mencionar el subsistema *Wine* que ofrece en los entornos *GNU/Linux* la posibilidad de ejecutar programas binarios del sistema *Windows*

[18]Una descripción detallada sobre la *Input Routine* se encuentra en [18]

[19]El programa ensamblador conocido como *Assembler PIG-2*

[20]Las principales *quickies* resuelven el cálculo de funciones trascendentales y el proceso de formato para la salida de datos, una lista completa se encuentra en el manual del programador de la máquina [15]

[21]El trabajo de digitalización del citado documento original se realizó a pedido del autor con la ayuda del *Centre for Research Collections* en la *Edimburgh University Library*

bastones largos [4] [5]. Sabemos que la máquina *Mercury* continuó en servicio hasta el año 1970 [19], pero con los elementos disponibles podemos concluir que es altamente probable que se dejara de utilizar el *COMIC* luego del exilio de sus creadores[22]. Menores son aún las posibilidades de que se hayan practicado modificaciones al mismo lenguaje y compilador. En síntesis estamos frente a un programa de computación que debido a las circunstancias propias del medio donde fue creado, quedó por decirlo de alguna manera bien gráfica, suspendido en el tiempo. Wilfred Durán a quien podemos considerar sin dudarlo como al padre del *COMIC*, tal vez anticipando las consecuencias del proceso político que se estaba dando en el país, realizó una copia del mismo el día 2 de Mayo de 1966. Cuando realizó esta copia de seguridad lo hizo descontando que dicho proceso que lo alejaba del desarrollo del *COMIC* llegaría a su fin en algún momento y entonces podría retomar el trabajo. Si bien el proceso llegó efectivamente a su fin, para ese momento la tecnología había cambiado y la continuidad del *COMIC* tenía poco sentido práctico. Wilfred conservó en su poder la cinta de papel perforado que contiene la copia maestra del *COMIC* hasta el mes de Mayo del año 2011. Hoy, continuar trabajando sobre el *COMIC* vuelve a tener sentido, obviamente sin perseguir utilidad práctica alguna sino atendiendo al alto valor histórico que podemos otorgarle como objeto de estudio sobre las técnicas utilizadas en el *IC* en aquellos años.

Hasta mediados de la década de 1960 la cinta de papel perforado constituía, junto con las tarjetas perforadas, uno de los medios más comunes para almacenar información. Vamos a resumir algunas de las características de esta tecnología que tienen relevancia a la hora de recuperar la información almacenada. La cinta de papel es un medio que podemos incluir en la clase *WORM* [23], la información se almacena realizando perforaciones en el papel, de manera que los cambios realizados son permanentes. Es importante destacar que no se trata exactamente de un medio de una sola escritura, existen dos técnicas para la corrección de errores y la realización de modificaciones en la información almacenada en cinta de papel. La primera consiste en la convención de utilizar como caracter de borrado aquel cuya representación consista solo de perforaciones [24]. La otra técnica es la que dio el nombre a las operaciones para edición de texto conocidas como Cortar y Pegar, claro que el trabajo era bastante mas laborioso que en la actualidad donde se utiliza simplemente una combinación de teclas. Se trata de un trabajo más artesanal que consiste literalmente en cortar la cinta de papel y empalmar, utilizando cinta adhesiva, nuevos tramos a fines de lograr las modificaciones deseadas. Analizar estas posibilidades de edición artesanal nos obliga a repasar otra de las características de la cinta de papel, su densidad. La densidad de la cinta de papel comparada con los medios magnéticos (incluidos los disponibles en los años sesenta del siglo XX) es baja, de hecho se requieren aproximadamente unos setenta centímetros lineales de papel para almacenar 256 caracteres. Algunos de los atributos expuestos motivaron la migración hacia otro tipo de medios de almacenamiento, fundamentalmente medios magnéticos. Notablemente son estos mismos los atributos que le confieren una clara ventaja cuando intentamos preservar los programas y datos almacenados. La baja densidad permite la lectura de la información por simple inspección visual y la larga vida útil del papel, no susceptible a campos magnéticos, cuando está bien conservado. [25]

Aún contando con estas ventajas a la hora de leer la información en forma manual, no contamos hoy con dispositivos para hacerlo en forma automática. Wilfred en su trabajo [9] deja abierta la pregunta sobre la existencia de alguna lectora de cinta de papel perforado. Es difícil asegurar que al inicio de este proyecto de restauración no existiera alguna lectora disponible. Es en cambio seguro que de existir alguna lectora en funcionamiento no tiene una exposición tal que nos permitiera dar con ella. Esta situación motivo la construcción de una lectora de cinta de papel perforado experimental. Para esto se partió de un antiguo equipo de teletipo *Siemens T1000* [26] del que se extrajo el módulo de la unidad lectora de cinta de papel. Luego se realizó un análisis del circuito controlador de este módulo a fines de comprender su funcionamiento y permitir así su conexión a un equipo de computación actual. Dicha comunicación se logró conectando distintos puntos del circuito electrónico original donde se encuentran disponibles los datos obtenidos por la lectora óptica. Desde estos puntos se cableó hacia una placa convencional de Entradas Salidas Digitales conectada a una computadora personal. Finalmente se anuló el avance motorizado de la cinta a fines de evitar posibles atascos de papel, eliminando los riesgo a la hora de procesar una cinta de la que solo se dispone una copia. [27]

Para completar el dispositivo lector de cintas se desarrolló un programa controlador que partiendo de las entradas digitales de la controladora genera la información codificada que se almacena posteriormente en un archivo de datos. De manera que las perforaciones en el papel se convierten en señales digitales que son interpretadas por el programa controlador y convertidas en un archivo de datos cuyo contenido es una representación de los datos almacenados en la cinta de papel. El carácter experimental de la lectora construida podría arrojar dudas sobre la validez de los datos obtenidos, cuestión que obligó a buscar formas de garantizar la fidelidad de los mismos. En un principio se consideró que la validación de los datos leídos se obtendría realizando múltiples lecturas de la cinta a fines de tener distintos archivos de datos para someter a un proceso de comparación que posibilitara detectar errores. Luego se logró comprender el algoritmo de validación de datos

[22] Wilfred Durán [9] asegura no haber dejado una copia del *COMIC* a los interventores del *IC*. Por otra parte existen testimonios de alumnos que se encontraban cursando sus carreras entre los años 1967 y 1970 indicando que en aquella época ignoraban la existencia de dicho lenguaje y compilador. (Según testimonio de Raúl Carnota, en conversaciones con el autor)

[23] WORM por sus siglas en ingles de Write Once, Read Many

[24] En el caso de la cinta de 5 canales, se utiliza el caracter 31, cuya representación en binario es una secuencia de cinco dígitos uno, lo que se codifica como todas las posiciones perforadas

[25] La cinta del *COMIC* con la que trabajamos se mantiene en perfecto estado de conservación desde hace 46 años

[26] La unidad lectora utilizada pertenecía a un teletipo *Siemens T1000* (fabricado entre los años 1976 y 1985). El equipo se encontraba en poder de un coleccionista particular residente de la localidad bonaerense de Pehuajó

[27] En rigor de verdad existen dos copias entregadas por Wilfred Durán a la FCEyN de la UBA, cantidad que no disminuye los cuidados necesarios para su tratamiento

(*checksum*) utilizado por la máquina original a la hora de generar la cinta. En la versión actual del programa controlador se implementó el mismo algoritmo de validación que consiste en el cálculo de la sumatoria módulo 1024, de los datos almacenados en cada tramo de la cinta. Haber implementado el mismo algoritmo en la nueva lectora, aún tratándose de un algoritmo falible [28] permite asegurar que los posibles errores se presentarían de idéntica forma en la máquina real. Para dar por terminada la tarea de transcodificación de la cinta se realizó una lectura adicional. Los datos obtenidos en ambas lecturas son idénticos.

III. Resultados preliminares

Si bien el proyecto se encuentra en etapa de desarrollo y con distinto grado de avance en cada uno de sus frentes, se pueden presentar y evaluar algunos resultados.

A. Programas de base

Luego de la transcripción del código fuente manuscrito hacia un archivo digital y utilizando el ensamblador cruzado que ya hemos presentado, se generó la imagen binaria de los dos primeros sectores (0 y 1) del tambor magnético virtual. Si bien el nivel de transcripción a digital del manuscrito de la *Input Routine* presenta un mayor avance, por el momento solo fue necesario trabajar con los dos primeros sectores. Los resultados obtenidos permiten pensar que es posible reconstruir la totalidad de la *Input Routine* utilizando este método de trabajo.

B. Emulador

El emulador de la *Mercury* tiene implementadas más del 70 % de las instrucciones máquina originales. En una primera etapa, que podemos definir como de factibilidad técnica, se trabajó implementando algunas instrucciones representativas de cada clase (lógicas, aritméticas, de salto, b-modificables, de punto flotante, de entrada salida, etc.) se desarrollaron para esta primera instancia pequeños programas que permitieron verificar el funcionamiento de la *arquitectura* del emulador. Concluida esta instancia el orden en la implementación de las instrucciones siguió la necesidad marcada por los programas disponibles, fundamentalmente los primeros sectores de la *Input Routine* y la secuencia de arranque del *COMIC*.

Aún con el emulador en etapa de desarrollo, la primera facilidad prestada por la *Input Routine* que se ejecutó con éxito, fue la que brinda soporte al procedimiento conocido como *Tele-output*. Este procedimiento permite obtener una copia en cinta de papel del contenido de los tambores magnéticos [29]. Como resultado se obtuvo en el archivo que hace las veces de cinta de papel de salida, una imagen del contenido de los tambores magnéticos que inicialmente tenían información solo en los sectores cero y uno. Esta imagen se analizó a fines de validar su correcto formato, identificando las metainstrucciones y datos de *checksum* agregados por la *Input Routine*.

C. COMIC

Utilizando el lector de cinta de papel experimental construido en el seno de este proyecto, se obtuvo una copia en formato digital de la cinta del *COMIC*. La lectura de la cinta demandó un gran cuidado ya que, como anticipamos, el sistema de arrastre de la lectora utilizada se desactivó a fines de proteger a la cinta frente a eventuales fallas mecánicas. El programa controlador de la lectora experimental validó el *checksum* de los 176 tramos o bloques de cinta que contienen al *COMIC*. Se documentó durante este mismo proceso los tramos de cinta agregados. Esto fue posible dado que la cinta casi en su totalidad es de papel color azul[30] sin embargo existen dos zonas donde aparece cinta de color rojo, tramos que fueron agregados con cinta adhesiva. Disponer de la información sobre el color de los distintos tramos es importante a la hora de estudiar su contenido. El archivo digital generado contiene como metadatos la información sobre el color de cada tramo, preservando así un detalle que de otra manera se perdería. La lectura de la cinta representa un hito importante desde el punto de vista de la preservación. independizando la conservación del *COMIC* del donde estuvo almacenado en forma exclusiva durante los últimos 46 años y validando a la vez la marcha del propio proyecto.

D. Primer arranque del ecosistema

Para facilitar la lectura, en las descripciones que siguen haremos referencia a los distintos dispositivos de la máquina por sus nombres sin indicar en cada caso que siempre se trata de sus representaciones virtuales. Se dispuso el contenido de los tambores magnéticos de la *Mercury* para que solo contuvieran los sectores de arranque de la *Input Routine* [31]. Se alimentó la unidad lectora de cinta con una copia del *COMIC*. En la consola de operación se seleccionó la secuencia de arranque correspondiente al proceso *Tele-input* [32], y luego se dio inicio a la secuencia[33]. En pocos segundos [34]la máquina se detuvo emitiendo la señal sonora que indica el final de la carga. Concluida la carga, se realizó la verificación del contenido de los tambores magnéticos y de la bitácora de actividad del emulador. Con esta información se pudo establecer en que sectores de los tambores magnéticos reside el *COMIC* al finalizar la carga. Este análisis mostró que la información de la cinta se almacena en distintas áreas de los tambores magnéticos ya que los sectores no son contiguos.[35] El análisis

[28] Si bien es poco probable, se pueden dar combinaciones de datos tales que leídos en distinto orden al que fueron escritos generen idéntica sumatoria o *checksum*

[29] El procedimiento *Tele-output* se invoca con la llave número 9 de la consola en posición encendido en el momento de lanzar el proceso *Initial Transfer*

[30] Según testimonio de Wilfred, en el *IC* reservaban las cintas de color azul para los compiladores *AUTOCODE* y *COMIC*

[31] Las funciones básicas para la lectura y escritura en los tambores magnéticos y la cinta de papel residen en los sectores 0 y 1

[32] La secuencia *Tele-input* se indica vía la llave de consola número 2

[33] El inicio se ordena mediante el pulsador *Initial Transfer*

[34] El tiempo de proceso no está expresado de acuerdo al requerido por la máquina real. El emulador permite ajustar la escala de tiempo de la máquina y sus periféricos

[35] Un mapa detallado se encuentra disponible en el sitio WEB del proyecto [17]

de las distintas áreas permite confirmar que la máquina del *IC* tenía en el momento de realizada la copia cuatro tambores magnéticos, ya que en la distribución de sectores existe un grupo que se aloja más allá del sector 512 que en la configuración de dos tambores oficia como barrera. Un hallazgo importante tiene que ver con la *Input Routine*. El proceso de carga del *COMIC* funcionó correctamente hasta donde se pudo analizar, leyendo la totalidad de la cinta, grabando y verificando la información en los tambores magnéticos y emitiendo la señal audible que indica el final de la fase de carga. Luego, durante el estudio del archivo de bitácora del emulador se presentó un hecho curioso: los primeros bloques leídos desde la cinta, fueron grabados por la *Input Routine* en los sectores cero y uno de los tambores magnéticos. Una vez finalizada la carga se verificó que el contenido de los sectores era el mismo que al inicio del proceso. Este simple hecho nos permite asegurar que los dos primeros bloques de la cinta no contienen una parte del *COMIC*, sino una parte de la *Input Routine* de la máquina del *IC* y esto nos permite confirmar que los sectores de arranque del entorno emulado coinciden con los de la máquina real aún sin contar con una prueba física del contenido ya que ,como se adelantó, los tambores no llegaron hasta nuestros días. Es probable que en la misma cinta se logren identificar otras partes de la *Input Routine* u otros programas que en el momento de generada la copia residían en los tambores magnéticos. Se espera que el estudio detallado del código del *COMIC* permita establecerlo con mayor precisión. Por otra parte, y en menor grado ya que son solo dos sectores, se pudo confirmar que el sistema de representación binario para instrucciones y datos [36] de la máquina real coincide con la interpretación que se hizo de la documentación disponible y a partir de la cual se desarrollaron el emulador y las utilidades de soporte empleadas en este proyecto de preservación. Una vez cargado en la máquina el *COMIC*, se pudo comenzar el estudio de las distintas opciones o secuencias de arranque del compilador, secuencias que no se encuentran documentadas en el manual [13]. Al momento se logró documentar las denominadas en el lenguaje de la máquina *Start Clear* y *Start No Clear*. La primera ejecutó una secuencia de instrucciones que inicializaron la memoria de trabajo antes de pasar a leer la cinta de entrada. Finalmente se elaboró un pequeño programa en lenguaje *COMIC* y se generó una cinta con su contenido. El compilador al concluir la etapa de inicialización de memoria de trabajo, entró en una fase que podemos denominar como carga del código fuente. En consecuencia se pudo observar como se procesó la cinta de entrada y se dispuso el contenido del programa fuente en distintas zonas de memoria de la máquina. Actualmente se está estudiando esta etapa que se puede entender como la primera fase en el trabajo de compilación.

IV. Conclusiones y próximos pasos

El propio método de trabajo seguido en este proyecto, consistente en la construcción de un *ecosistema* donde ejecutar antiguos programas de computación, sin duda se ha enriquecido con esta primera puesta en práctica. De forma que las tareas realizadas en la práctica y requeridas por el proyecto de preservación del *COMIC* contribuyeron a poner de manifiesto las distintas técnicas y herramientas postuladas en el método descripto.

Desde el punto de vista técnico, nos encontramos en un punto donde la factibilidad está probada, situación que habilita el desarrollo completo del emulador y sus herramientas de soporte a fines de producir las primeras versiones para distribución.

La etapa denominada *Adquisición de experiencia*, podrá comenzar en tanto se cuente con la posibilidad de distribuir el emulador junto con los programas de base (*Input Routine*) y el mismo *COMIC*. Es esta ultima la etapa más interesante del proyecto, ya que es donde se esperan las mayores contribuciones tanto de las personas que han trabajado en el *IC* como de otras que interesadas por las cuestiones históricas y técnicas se quieran involucrar de forma directa trabajando en el entorno recreado. Esta es una de las formas en las que se espera que el *ecosistema* inicial, el de carácter técnico, de paso a uno más amplio que permita nuevos estudios o investigaciones.

Disponer del *ecosistema* técnico materializado en el entorno de emulación que incluye los programas preservados, habilita futuros estudios sobre cuestiones de índole no solo técnica sino también del entorno de trabajo. Algunos de los protagonistas podrían volver a tomar contacto con un entorno de trabajo que ya no existe en forma física. En forma paralela, los interesados en comprender en detalle las soluciones ideadas en el *IC* durante sus primeros años de existencia, podrían hacerlo utilizando una máquina de idénticas características lógicas capaz de ejecutar los programas originales, este hecho le confiere al análisis una perspectiva de privilegio sobre el análisis estático que como vimos sería producto de la lectura de programas y documentación. Tratar de dar respuesta a los interrogantes que se planteaban en el *IC* entre los años 1960 y 1966, hacerlo desde la actualidad, es un ejercicio que permite una mayor compresión del *ecosistema ampliado* que constituyó el Instituto de Cálculo.

Como anticipamos este es un proyecto, que si bien ha dado sus primeros frutos, se encuentra en un estado de evolución inicial. Está previsto en próximas etapas: el desarrollo y publicación de la versión inicial del emulador y su conjunto de herramientas de soporte, la publicación de la documentación y programas preservados, la elaboración de nueva documentación para facilitar el trabajo de quienes se quieran iniciar en estas tecnologías, el estudio del código binario del compilador y la investigación específica sobre el dispositivo de graficación (*plotter*) utilizado en el *IC* y para el cual el *COMIC* brinda soporte.

Agradecimientos

El autor quiere agradecer a los organizadores de las Jornadas Sadosky[37] que fueron el disparador y fuente de motivación

[36] Instrucciones y datos presentes en los primeros sectores de la *Input Routine*

[37] Un resumen de las Jornadas Sadosky realizadas los días 12 y 13 de Mayo de 2011, en la Facultad de Ciencias Exactas y Naturales de la Universidad de Buenos Aires, se encuentra disponible en [20]

para este proyecto de preservación y estudio, a quienes fueron pioneros de la computación en la Argentina y que de distintas formas contribuyeron con sus testimonios y apoyo: Victoria Bajar, Cristina Zoltan, Liana Lew, Noemí García, Wilfred Durán, Jonás Paiuk y Ernesto García Camarero, a otras personas que aún sin tener relación directa con el proyecto brindaron su ayuda: Jack Ponton por su trabajo para que se pudiera disponer del documento *Anotatted Input Routine*, a Ariel Palazessi y Ricardo Garcia por su ayuda con el lector de cinta experimental. Particularmente a Raúl Carnota quien alentó este proyecto desde el primer momento aportando su inestimable ayuda y estímulo.

REFERENCIAS

[1] I. P. Pérez and C. A. M. López, "Introducción a la arqueología informática," 2011. [Online]. Available: www.uaeh.edu.mx/nuestro_alumnado/icbi/articulos/introduccion_arq_inf.pdf

[2] A. S. Chacur and V. Bajar, "Picme (programa interpretativo de ceuns por mercury)," 1963.

[3] R. Carnota and R. Rodriguez, *Fulgor y Ocaso de CEUNS. Una apuesta a la tecnología nacional en el Sur de Argentina*. Universidad Nacional de Rio Cuarto, 2011. [Online]. Available: www.cos.ufrj.br/shialc/content/docs/2.1_30SHIALCCarnota_Paper.v2.pdf

[4] F. Pigna and M. Seoane, *LA NOCHE DE LOS BASTONES LARGOS*, E. C. y Caretas, Ed. Fundación Octubre, 2006.

[5] E. D. de Guijarro, "1966: La noche de los bastones largos. el final de una etapa," *La Ménsula*, 2008. [Online]. Available: digital.bl.fcen.uba.ar/Download/002_LaMensula/002_LaMensula_006.pdf

[6] M. Sadosky, "Cinco años del instituto de cálculo de la universidad de buenos aires," *Ciencia Nueva*, 1972. [Online]. Available: www.cienciaanueva.com/documentos/CIENCIANUEVA19.pdf

[7] C. Mantegaril, "El instituto de cálculo de la uba [1957-1966]: La vigencia de un símbolo," *Ciencia Hoy*, 1995. [Online]. Available: www.cienciahoy.org.ar/hoy29/calculo01.htm

[8] P. M. Jacovkis, "Un lugar para clementina. el instituto de cálculo entre 1957 y 1966," *La Ménsula*, 2011. [Online]. Available: digital.bl.fcen.uba.ar/Download/002_LaMensula/002_LaMensula_013.pdf

[9] W. O. Duran, C. Zoltan, L. S. Lew, C. D. Cortes, and N. S. García, *Historia de la Informatica en Latinoamerica y el Caribe:Investigaciones y testimonios*. Universidad Nacional de Rio Cuarto, 2009, ch. 7. COMIC: El primer lenguaje y compilador argentino, desarrollado en el Instituto de Cálculo en 1965.

[10] D. E. Knuth and L. T. Pardo, *Selected Papers on Computer Languages*. Center for the Study of Language and Information, 2003, ch. 1. The Eearly Development of Programming Languages.

[11] R. A. Brooker, "The autocode programs developed for the manchester university computers," *The Computer Journal*, 1958.

[12] ——, "Further autocode facilities for the manchester (mercury) computer," *The Computer Journal*, 1958.

[13] W. O. Duran, *INTRODUCCION AL LENGUAJE COMIC*. Peru 272 - Buenos Aires - Argentina: Instituto de Cálculo, Mayo 1965.

[14] T. Brooker, "Mercury and its autocode," *Computer Resurrection*, 2011/2012. [Online]. Available: www.cs.man.ac.uk/CCS/res/res56.htm

[15] *FERRANTI MERCURY COMPUTER PROGRAMMERS' HANDBOOK*, Ferranti Limited Computer Department, Buenos Aires, April 1960.

[16] E. G. Camarero, *INTRODUCCION AL SISTEMA DE PROGRAMACION CONVENCIONAL PARA LA COMPUTADORA MERCURY*, Buenos Aires, 1962. [Online]. Available: http://elgranerocomun.net/Introduccion-al-Sistema-de.html

[17] G. del Dago. (2012) Mercury 20. [Online]. Available: http://mercuryemulator.blogspot.com.ar

[18] J. A. Fotheringham and M. de V. Roberts, "An input routine for the ferranti mercury computer," *The Computer Journal*, 1958.

[19] R. Carnota and M. O. Perez, *Historia de la Informatica en Latinoamerica y el Caribe:Investigaciones y testimonios*. Universidad Nacional de Rio Cuarto, 2009, ch. 8. Continuidad formal y ruptura real: la segunda vida de Clementina.

[20] (2011) Celebración del cincuentenario de la puesta en funcionamiento de clementina. [Online]. Available: www.dc.uba.ar/events/cincuenta