UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Arqueología informática: la calculadora programable Programma 101

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Antonio Estrella Contreras

Director: Xavier Molero Prieto

29 de julio de 2015

A Ana, David y todos aquellos familiares y amigos que me han ayudado y apoyado.

Resumen

La Olivetti Programma 101 fue presentada en sociedad en la Exposición Universal de Nueva York de 1965 y empezó a ser comercializada ese mismo año. Para entonces fue un gran avance tecnológico, tanto, que llegó a considerarse el primer ordenador personal de la historia. Su utilización se extendió en el ámbito doméstico y empresarial. Fue utilizada en proyectos de gran envergadura de la NASA y en la guerra de Vietnam. Este éxito finalizó cuando la empresa Hewlett Packard sacó al mercado su producto HP 9100 basado en todos los aspectos en la Programma 101, desbancándola del mercado.

En este trabajo, además de un estudio histórico, se realiza un análisis de la arquitectura y programación de este ordenador, al tiempo que se elaboran comparaciones con un procesador con arquitectura RISC. Dado el gran valor histórico de la Programma 101, parte de este trabajo ha servido para desarrollar una página web destinada al Museo de Informática de la Escuela Técnica Superior de Ingeniería Informática de la UPV y contribuir así a la difusión del patrimonio digital.

Palabras clave: Olivetti Programma 101, ordenador personal, MIPS R2000, arqueología informática, Museo de Informática.

Resum

La Olivetti Programma 101 va ser presentada en societat en l'Exposició Universal de Nova York de 1965 i va començar a ser comercialitzada aqueix mateix any. Per a llavors va ser un gran avanç tecnològic, tant, que va arribar a considerar-se el primer ordinador personal de la història. La seua utilització es va estendre en l'àmbit domèstic i empresarial. Va ser utilitzada en projectes de gran envergadura de la NASA i en la guerra de Vietnam. Aquest èxit va finalitzar quan l'empresa Hewlett Packard va traure al mercat el seu producte HP 9100 basat en tots els aspectes en la Programma 101, desbancant-la del mercat.

En aquest treball, a més d'un estudi històric, es realitza una anàlisi de l'arquitectura i programació d'aquest ordinador, al mateix temps que s'elaboren comparacions amb un processador amb arquitectura RISC. Donat el gran valor històric de la Programma 101, part d'aquest treball ha servit per a desenvolupar una pàgina web destinada al Museu d'Informàtica de l'Escola Tècnica Superior d'Enginyeria Informàtica de la UPV i contribuir així a la difusió del patrimoni digital.

Paraules clau: Olivetti Programma 101, ordinador personal, MIPS R2000, arqueologia informàtica, Museu d'Informàtica.

Abstract

The Olivetti Programma 101 first appeared in the Universal Exhibition of New York in 1965 and it was in the market the same year. By then, it was a great technological advantage, it was consider as the first personal computer of all time. Its usage spread in a domestic scope as well as in a business one. It was used in NASA large projects and in the Vietnam War. Its succeed finished when the Hewlwtt Packard released the HP 9100, completely based on the Programma 101, supplanting it of the market.

This project is a historic study as well as it is a deep analysis of the archeology and the programming of this computer, and at the same time, a comparation of a processor with structure RISC is made. Taking into account the historic value of the Programma 101, part of this project has been useful to develop a web page to the Museum of Informatics of the School of Informatics - iSchool at the UPV, and it contributed to the digital heritage spreading.

Keywords: Olivetti Programma 101, personal computer, MIPS R2000, computer archeology, Museum of Informatics.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

Este trabajo fin de grado está enmarcado dentro de lo que se denomina «arqueología informática», y el objeto de estudio de este es la Olivetti Programma 101. Esta máquina es considerada el primer ordenador personal de la historia. Para comenzar, en este capítulo se desarrolla una descripción tanto de la motivación de este trabajo fin de grado, como una enumeración de los objetivos que se pretenden conseguir con él.

1.1 Motivación

La arqueología informática [6] consiste en el estudio del software, hardware y todo aquello relacionado con el objeto de estudio como pueden ser la época en la que se desarrolló, sus creadores, las ventajas e inconvenientes que proporcionaba o las razones que impulsaron su creación.

La arqueología informática también recibe el nombre de arqueología computacional, informática clásica o retroinformática y sirve para investigar, analizar y comprender el diseño y funcionamiento del material informático; de esta forma se tiene la posibilidad de dar aplicación a los conocimientos adquiridos en futuros diseños y de paso enriquecer la cultura informática.

Las actividades de la arqueología informática se pueden clasificar en dos categorías según el ámbito en las que se aplica. Por un lado, en el ámbito tecnológico tenemos las actividades de simulación y desarrollo de software y hardware. Por otro lado, en el ámbito cultural y social nos encontramos con las actividades de divulgación, preservación, exposición, documentación, coleccionismo, congresos, asociacionismo y comercio.

El estudio en arqueología informática sigue unos pasos aproximados a los que se mencionan a continuación, aproximados porque pueden verse modificados según las necesidades del estudio en cuestión:

1. Definir y detallar lo máximo posible el objeto de estudio.
2. Investigar, reunir, analizar y organizar información sobre personajes, dispositivos físicos, soportes lógicos con información y documentos sobre el elemento definido.
3. Definir la secuencia y relaciones existentes entre la información recopilada en los puntos anteriores.
4. Crear la documentación correspondiente que se utilizará como fuente de información para futuras consultas.

De la arqueología informática no solo se obtiene información histórica, sino que además, recopilamos conocimientos de la estructura, diseño y funcionamiento del objeto de estudio, no olvidemos que en este trabajo es la Programma 101, la cual se puede ver en la figura 1.1. Estos conocimientos que obtenemos son lo que se denomina «patrimonio cultural» y en el caso del patrimonio generado en el ámbito de la informática se llama «patrimonio digital».

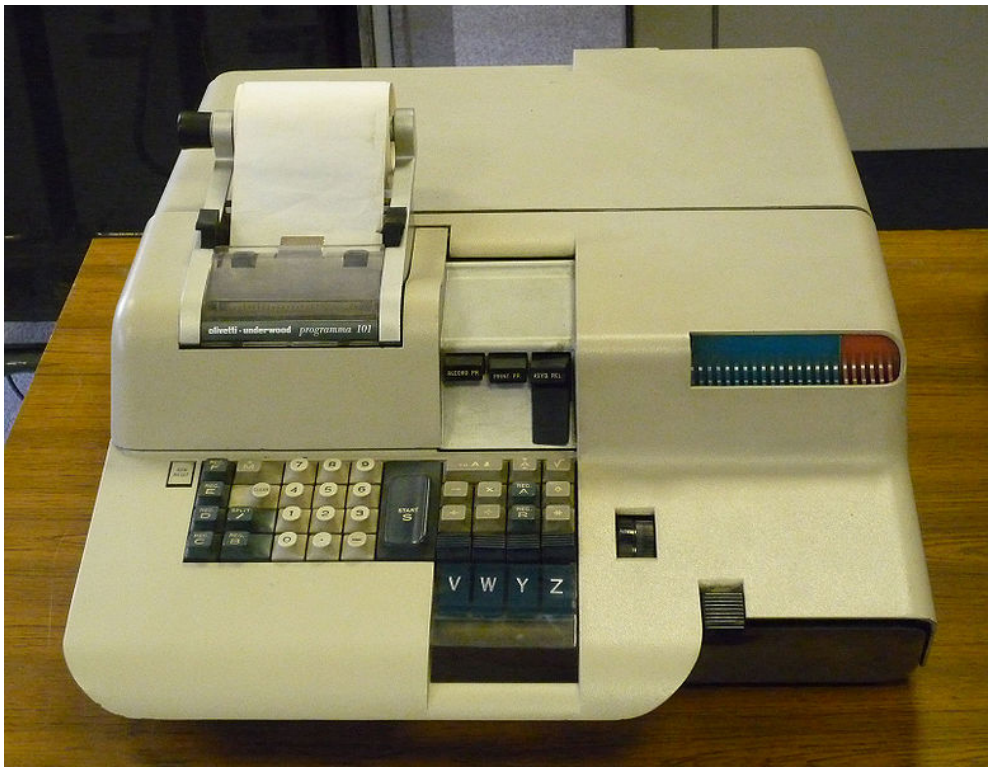


Figura 1.1: La Olivetti Programma 101 [17], objeto de estudio de este trabajo.

Cuando hablamos de patrimonio, patrimonio cultural o patrimonio de cualquier otra índole, nos vemos obligados a hablar de la UNESCO [23] (Organización de la Naciones Unidas para la Educación, la Ciencia y la Cultura) que desde su creación en 1945 vela por la paz en el mundo difundiendo y protegiendo la educación, la ciencia y la cultura. Su bandera (y logotipo) se muestra en la figura 1.2.

Figura 1.2: Bandera de la UNESCO que incorpora su logotipo [23].

En la UNESCO [23] han establecido las definiciones de los términos *patrimonio* como «[...] nuestra herencia del pasado, nuestros bienes actuales y lo que legamos a las generaciones futuras. El patrimonio es, o debería ser, algo que se transmite de generación en generación porque se valora», *patrimonio cultural* como «[...] aquellos lugares y objetos tangibles e intangibles que poseen valor cultural, histórico, estético, arqueológico, científico, etnológico o antropológico para determinados grupos o individuos [...]» y *patrimonio digital* tal como sigue:

> El patrimonio digital está formado por los materiales informáticos de valor perdurable dignos de ser conservados para las generaciones futuras, y que proceden de comunidades, industrias, sectores y regiones diferentes. No todos los materiales digitales poseen valor perdurable, pero los que lo tienen exigen metodologías de conservación activas para mantener la continuidad del patrimonio digital.

Como la propia UNESCO indica, en el mundo se generan ingentes cantidades de información de origen digital en diversos formatos, por lo que se debe recopilar y proteger como patrimonio cultural que es, con toda su importancia y consecuencias que implica, elaborando políticas y técnicas concretas. Para ello y ante la enorme importancia que presenta el patrimonio digital para la humanidad, la UNESCO redactó una carta para la preservación del patrimonio digital [23], la cual se puede leer en el apéndice A.

Esta carta está formada por un preámbulo y doce artículos repartidos en cuatro apartados. En ella se define con más detalle el concepto de patrimonio digital y su importancia en sí misma. Además, nos habla sobre la preservación de este tipo de patrimonio, las medidas necesarias para ello y las atribuciones que los estados miembros deberían cumplir. Respecto a este último punto hay que destacar la importancia de tres ideas que se encuentran en el artículo décimo de esta carta:

- Colaborar con museos, archivos y bibliotecas nacionales en la preservación del patrimonio digital.
- Fomentar la investigación y formación, así como impulsar la difusión de conocimientos y experiencias.
- Promover que instituciones de investigación y universidades velen por conservar los datos procedentes de investigaciones.

El rastro que nos deja el patrimonio digital en el transcurso de los tiempos y más concretamente el hardware y el software, nos enseña la evolución que ha sufrido el mundo de la informática desde sus inicios hasta la actualidad. De esta evolución se puede aprender, desde diversas perspectivas, cómo el ser humano se las ha ideado para solucionar mediante la informática los retos que se le planteaban en el día a día y en cada vez más ámbitos de la sociedad.

Como la informática está involucrada en todos los aspectos de la vida, su historia es tan importante como la historia general y todo el mundo debería tener una mínima educación en este campo. De esta forma se valoraría más la informática de la actualidad, ya que se sabría qué ideas, dificultades, errores y logros han sido necesarios para llegar a ella.

Todo aquello que nos enseña la historia de la informática en manos de los actuales y futuros ingenieros informáticos toma una especial relevancia, puesto que, como expertos en la materia que ya son o que llegarán a ser, podrán darle un sentido más constructivo a la información histórica, dar utilidad a puntos de vista que posiblemente ya estén descartados pero que con las nuevas tecnologías podrían volver a ser aprovechados, aprender de los logros y errores; en definitiva, la historia de la informática sirve como herramienta para el I+D+I en este ámbito.

1.2 Objetivos

A partir de un objeto de estudio, que como se ha mencionado en la primera página de este documento, se trata de la Olivetti Programma 101, en este trabajo se pretende alcanzar los objetivos que se enumeran a continuación:

1. Se procede a situar a la Programma 101 dentro de un marco histórico para, de esa forma, hacernos una idea de en qué circunstancias tanto técnicas, empresariales como sociales, surgió esta máquina. Además se explica por quién, cómo y dónde fue desarrollada y presentada al público, y se muestra un panorama de la repercusión que tuvo su aparición tanto en la sociedad como en las empresas de aquella época.
2. Se analiza la arquitectura interna, el diseño y funcionamiento de la Programma 101 haciendo hincapié en el tipo de electrónica, memoria, novedades tecnológicas y en la facilidad de manejo mediante una específica organización de

registros y un reducido juego de instrucciones. Con ello se intenta denotar el nivel de dificultad que supuso el desarrollo de la Programma 101 para sus diseñadores, sin tener ninguna referencia previa.

3. A continuación, se pretende mostrar la facilidad de programación de la Programma 101 diseñando y comentando programas de diversa índole utilizando su juego de instrucciones. Para ello, se han seleccionado unos algoritmos en los que el programador necesita hacer uso de las diversas funciones y características que ofrece esta máquina.

4. Se compara la arquitectura interna, funcionamiento y programación de la Programma 101 con el procesador MIPS R2000. Este se utiliza en la actualidad con propósito pedagógico en las asignaturas que se imparten en este centro relacionadas con la estructura y la arquitectura de computadores dentro de la titulación de Grado en Ingeniería Informática. Esta comparativa nos proporciona una referencia para dar una visión más concreta de cómo estaba formada, funcionaba y se programaba con la Programma 101.

5. La mayor parte de la información que se puede encontrar sobre la Programma 101 está en Internet, donde existen múltiples sitios dedicados a dicha máquina, ya que esta cuenta con un gran número de admiradores. Entre dichos admiradores hay personas que han desarrollado simuladores de esta.

6. Finalmente, formando parte de este trabajo y como colaboración con el proyecto de difusión cultural que está llevando a cabo el Museo de Informática, se han diseñado dos páginas web dedicadas a la Programma 101 —una en castellano y otra en valenciano— con carácter didáctico, para de esta forma, intentar darle toda la difusión posible y situarla en el lugar de la historia que le corresponde.

1.3 Estructura de la memoria

Para el desarrollo de esta memoria se ha dividido su contenido en nueve capítulos y cuatro apéndices, entre todas estas secciones, se ha expuesto la información necesaria para cubrir los objetivos mencionados en el apartado anterior. A continuación se explica como se ha estructurado este trabajo:

1. Se ha realizado una introducción exponiendo; los motivos que originan la creación de este trabajo; los objetivos que pretende alcanzar; la estructura que se explica en este apartado; el uso del material bibliográfico que se ha realizado durante su desarrollo y los agradecimientos a las personas y entidades que han colaborado.

2. Tras una breve descripción sobre las generaciones en el mundo de las computadoras, se relata; cuándo, cómo y quién desarrolló la Programma 101; su

presentación, repercusión y aplicación de esta en la sociedad de aquella época y la situación empresarial que se vivía.

3. Se explica la arquitectura interna dando detalles del tipo de electrónica utilizada, la memoria ideada para este fin y las novedades como: las tarjetas magnéticas —y su correspondiente lector/grabador— y su estética diseñada expresamente para este desarrollo.

4. Procedemos a enseñar cómo está organizada la memoria y los registros, para fundamentar así, la descripción del juego de instrucciones —dividida según sus propósitos— y la introducción de constantes en tiempo de ejecución. Se acompaña esta sección con una comparativa de las características mencionadas de la Programma 101 con las del procesador MIPS R2000.

5. Se diseñan y comentan seis algoritmos; cinco funciones matemáticas, un simulador de aterrizaje lunar —la NASA (*National Aeronautics and Space Administration*) utilizó la Programma 101 con un programa de este tipo para la misión Apolo 11— y el cálculo de la órbita de un satélite alrededor de la Tierra. Este último fue mostrado en la presentación oficial al público en la Exposición Universal de Nueva York en 1965 y causó gran expectación. En esta sección también se realizan comparaciones con la programación del MIPS R2000.

6. Una vez se han descrito los aspectos técnicos de la Programma 101, se realiza lo propio para la calculadora programable de Hewlett Packard HP 9100, al mismo tiempo que se compara con la Olivetti. De esta forma podemos comprobar las semejanzas existentes entre ambas y entender porque la empresa norteamericana pagó 900 000 dólares a la italiana en concepto de regalías.

7. Se enumeran y comentan los recursos en Internet más importantes consultados durante la realización de esta memoria y que versan sobre la Programma 101. En esta relación, se incluyen las referencias a sitios web de desarrolladores de simuladores. De entre todos los simuladores que podemos encontrar se han seleccionado dos para su descripción y aprendizaje de funcionamiento y así, con su uso, se pueden hacer demostraciones reales de su manejo.

8. Con la información recopilada en este trabajo y después de hacer una presentación del Museo de Informática, se procede a diseñar dos páginas para su sitio web. En esta sección, se detalla su estructura y el tipo de información que aporta —en formato texto, imagen y audiovisual— para darle toda la difusión posible a la Programma 101 y situarla en el lugar de la historia que le corresponde.

9. Exponemos las conclusiones que se alcanzan con este trabajo y que han surgido a partir de los objetivos descritos en el primer capítulo. También se hacen unas sugerencias de posibles futuros proyectos que han ido apareciendo en el transcurso de esta memoria.

10. Por último, se añaden unos apéndices donde se puede consultar información adicional como: la carta para la preservación del patrimonio digital —UNESCO—, la patente de la Programma 101, una plantilla de programación y una relación de esta cumplimentada con las seis programaciones correspondientes a los algoritmos descritos en este documento.

1.4 Uso del material bibliográfico

Como se puede comprobar en el apartado de bibliografía y en las referencias a pie de página que se pueden encontrar a lo largo de este documento, se ha consultado un elevado número de material bibliográfico. El origen de todos ellos se puede clasificar en: libros, monografías, artículos en revistas, manuales técnicos del fabricante, enlaces a sitios web, la patente de la Programma 101 e incluso varios vídeos sobre esta.

La recopilación de todo este material bibliográfico ha sido un trabajo dividido en tres fases; localización, análisis y aceptación, es decir, se ha tenido que buscar la información, comprobar si esta es válida para el propósito buscado y si no ha sido rechazada, se guarda para su uso en este documento.

Entre los libros y monografías, nos podemos encontrar con material muy diverso, desde los altamente técnicos como el «Estructura y diseño de computadores: la interfaz software/hardware» [4] y «Organización y diseño de computadores: la interfaz hardware/software» [7], hasta los cercanos al género novelesco como el escrito por Pier Giorgio Perotto; «Programma 101 - L'invenzione del personal computer: Una storia appassionante mai raccontata» [27]. Tanto unos como otros han aportado información valiosa para el desarrollo de este documento.

Unas de las fuentes de información más abundantes e importantes se encuentran en los recursos localizados en Internet, por ello se dedica el capítulo 7 expresamente a estos. Además de los encontrados en el apartado de la bibliografía y en el capítulo mencionado, se pueden encontrar a pie de página más referencias, ya que puntualmente se han consultado o se han obtenido imágenes. A través de Internet —mediante correo electrónico— se han obtenido; permisos de uso [22], simuladores [10, 3] e información adicional muy útil en esta memoria.

Otro recurso obtenido de Internet es la patente de la Programma 101 [28], donde se puede consultar sus especificaciones y datos técnicos. Ha sido obtenida gracias al servicio que proporciona la empresa Google para el acceso a multitud de patentes.

Respecto a los manuales del fabricante —Olivetti—, se han manejado cuatro de ellos; una biblioteca de programación —el volumen 1—, manual de programación, general, así como una referencia [18, 19, 20, 21]. En la figura 1.3 se muestran las portadas de estos.

Figura 1.3: Portadas de los manuales de Olivetti utilizados como material bibliográfico.

Por otro lado, se han seleccionado dos artículos de la revista de retroinformática «Jurassic News» [11, 9] donde se habla de los dos simuladores seleccionados y que se describen en el capítulo 7. Las portadas de estas se pueden observar en la figura 1.4.

Figura 1.4: Portadas de las revistas «Jurassic News» utilizadas como material bibliográfico.

Y por último, el material bibliográfico en formato audiovisual ha sido de gran ayuda en la elaboración de este trabajo, concretamente han sido dos; el documental «Programma 101: La máquina que cambió el mundo» [31] y el vídeo de archivo «Italia - Presentato il calcolatore elettronico da tavolo» [1].

1.5 Agradecimientos

Para el desarrollo de este trabajo se ha mantenido conversación a través de correo electrónico con la *Associazione Archivio Storico Olivetti* para pedir permiso de uso del material expuesto en su web. Por ello, agradezco a Lucia Alberton como interlocutora de la asociación las gestiones que ha realizado para la obtención de dicho permiso.

Respecto a los simuladores, agradezco a Marco Galeotti su simulador que nos ha sido amablemente cedido a través de correo electrónico. También doy las gracias al ingeniero Claudio Larini, ya que tras ponerme en contacto con él a través de correos electrónicos, he recibido por su parte una gran disponibilidad, atención y colaboración en este trabajo así como la cesión de su simulador, información sobre él y los programas que integra.

Finalmente, mi más sincero agradecimiento al director de este trabajo fin de grado, Xavier Molero Prieto, profesor del DISCA (Departamento de Informática de Sistemas y Computadores) y director del Museo de Informática, el cual me ha introducido y orientado en la temática de este documento, así como, la tutela y la corrección del proyecto tanto desde el punto de vista estructural, como de implementación.

CAPÍTULO 2

Contexto histórico

En este capítulo se describe el momento en la historia de la informática en el que apareció la Programma 101, por quién y cómo fue desarrollada, cuándo y dónde fue presentada al público y su repercusión en la sociedad de aquella época. Además, se explican las diversas situaciones en las que se encontraron tanto la empresa Olivetti como el equipo de desarrollo ante las circunstancias empresariales del momento.

2.1 Panorama histórico: las generaciones

La evolución de la informática está estrechamente ligada a la evolución que ha sufrido el hardware en el transcurso de la historia, ya que a medida que han surgido nuevos componentes, la informática ha podido aprovechar las ventajas y funcionalidades de estos para poder mejorar.

La informática está clasificada históricamente en cinco generaciones [7] según el punto de la evolución de la tecnología del hardware en el que se encuentra. Según unas fuentes u otras las fechas varían e incluso el número de estas, algunos hasta añaden una sexta generación.

- **Primera generación (1950-1959):** La tecnología de las computadoras estaba basada en los tubos de vacío, unos componentes que necesitaban un gran consumo de energía, disipaban mucho calor, eran de tamaño considerable y su duración era muy limitada. Los computadores construidos con este tipo de componentes —por ejemplo UNIVAC I en 1951 e IBM 650 en 1953— eran muy grandes y pesados. A mediados y finales de esta generación se hace necesario la aparición de los primeros lenguajes de programación de alto nivel como el FORTRAN (1954), ALGOL (1958), LISP (1958) y COBOL (1959).
- **Segunda generación (1960-1968):** La aparición de un nuevo componente electrónico llamado transistor revolucionó el desarrollo de nuevos compu-

tadores —por ejemplo IBM 1401 en 1960 y ATLAS en 1962— reduciendo el consumo de energía, el tamaño y por tanto el peso, así como un aumento en prestaciones como la capacidad de memoria y la velocidad. Este desarrollo está acompañado con la aparición de nuevos lenguajes de programación de alto nivel más avanzados como el BASIC (1964).

- **Tercera generación (1969-1977):** La tecnología evolucionó de tal forma que apareció el circuito integrado (chip). Fue desarrollado por IBM —utilizado en la construcción del IBM 360 en 1964 y del DIGITAL PDP-11 en 1970— y contenía gran cantidad de transistores y otros componentes electrónicos discretos en una pequeña oblea de silicio. De esta forma se conseguirían mejorar aún más las ventajas que se obtuvieron en la generación anterior. El avance alcanzado en esta generación promovió la industria del software e hizo evolucionar los sistemas operativos.

- **Cuarta generación (1978-199?):** El poder de integración que se consiguió con los circuitos integrados gracias a las tecnologías LSI (*Large-Scale Integration*) y VLSI (*Very-Large-Scale Integration*) llevó a la creación por parte de la compañía norteamericana INTEL del primer microprocesador —INTEL 4004—. Este chip integraba, por primera vez, toda la electrónica necesaria para constituir bancos de registros, una unidad de control (UC) y una unidad aritmético/lógica (ALU), entre otros elementos.

- **Quinta generación (199?-):** Para algunos autores esta última generación no existe, ya que no se enmarca en la clasificación por la evolución de la tecnología utilizada, sino por el uso que se da a la informática en los ámbitos del procesamiento paralelo, interconexión de computadores —redes—, inteligencia artificial, multimedia, entre otros.

Esta clasificación nos sirve para enmarcar a la Programma 101 dentro de la segunda generación, ya que surgió en 1964 y su construcción estaba basada en componentes discretos, siendo los transistores el más importante de ellos. En esa misma década, los computadores seguían siendo muy grandes, incluso el IBM 360 que ya era una innovación por estar construido con circuitos integrados (tercera generación), eran de un tamaño considerable y dirigidos a unos usuarios muy especializados, todo lo contrario que la Programma 101.

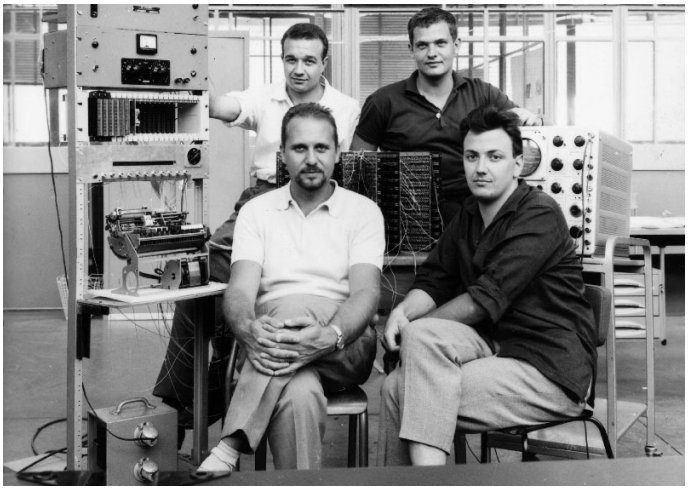
2.2 Desarrollo y presentación

En la primavera de 1962 Roberto Olivetti, el entonces presidente de la compañía italiana Olivetti, dio instrucciones al ingeniero Pier Giorgio Perotto (1930-2002) para iniciar el estudio de viabilidad de una calculadora electrónica capaz de automatizar una secuencia de instrucciones, al alcance de un usuario no especializado, a un

precio razonable y con unas dimensiones reducidas parecidas a las de una máquina de escribir.

Pier Giorgio Perotto ideó una máquina pequeña, flexible, con memoria, fácil de usar y programable mediante un lenguaje de programación sencillo para que cualquier persona pudiera programarla, básicamente un ordenador personal. Tal y como escribió más tarde [27], una máquina que no solo destaque por su velocidad o potencia, sino más bien por su autonomía funcional, textualmente «*[...] una macchina nella quale non venga solamente privilegiata la velocità o la potenza, ma piuttosto l'autonomia funzionale, [...]*» —Pier Giorgio Perotto, 1995, "Programma 101".—

Para desempeñar el encargo, Perotto creó un equipo de desarrollo con cuatro hombres más [22, 29]; Gastone Garziera, Giovanni De Sandre, Giancarlo Toppi y Giuliano Gaiti, estos junto con el segundo prototipo de la Programma 101 aparecen en la figura 2.1(a) menos Gaiti, que fue quien realizó la fotografía con la cámara de Perotto en agosto de 1964.



(a) Equipo de desarrollo de la Programma 101.[1]

(b) Gastone Garziera.[2]

(d) Giancarlo Toppi.[3]

(c) Pier Giorgio Perotto.[4]

(e) Giovanni De Sandre.[5]

Figura 2.1: Equipo de desarrollo de la Programma 101 menos Giuliano Gaiti. Izquierda: Agosto de 1964. Derecha: Recientemente (mismo orden en ambas partes).

El proyecto comenzó con unas especificaciones muy claras, pero la falta de una referencia de partida, debido a que nunca se había hecho nada parecido, añadió una gran dificultad en su realización. A esta falta de referencia se le añadía la prisa en que se debía realizar el proyecto, ya que no sabían si la competencia estaba trabajando en algo parecido y era muy importante que la Programma 101 saliera al mercado antes que cualquier otro posible producto similar.

[1] Imagen cortesía de Associazione Archivio Storico Olivetti, Ivrea, Italia. [22]
[2] `http://www.p101.it/p101-tells-p101`
[3] `https://www.facebook.com/giancarlo.toppi.7`
[4] Imagen cortesía de Associazione Archivio Storico Olivetti, Ivrea, Italia. [22]
[5] `http://www.parlarealmicrofono.it/tag/giovanni-de-sandre`

Dos puntos muy importantes para conseguir el éxito del proyecto fueron, por un lado, sustituir las memorias existentes en el momento por memorias de línea de retardo acústica magnetostrictiva; era una buena opción por su tamaño, sencillez y coste porque la empresa estaba capacitada para su fabricación. Por otro lado, se utilizaron los nuevos —en aquel momento— transistores 2N708 que, junto a su reducido tamaño, no tenían tantos problemas de sobrecalentamiento y fiabilidad como las válvulas de vacío que se utilizaban hasta el momento, además, su vida útil era excelente.

El trabajo del equipo de desarrollo de la Programma 101 finalizó su tarea en abril de 1964 —dos años de trabajo— consiguiendo un producto que sería el comienzo de la historia de los PC (*Personal Computer*) u ordenadores personales y que bautizaron, por motivos de mercadotecnia, con un nombre femenino: «Programma 101».

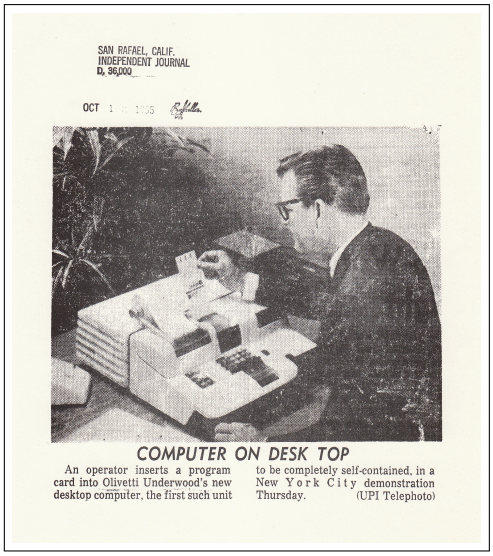
La Programma 101 necesitaba ser mostrada al público para hacer creíbles sus características y potencial. Por ello, en octubre de 1965 tuvo la ocasión de darse a conocer en la Exposición Universal de Nueva York, dentro de la sección dedicada a las empresas que querían presentar sus productos llamada BEMA (*Business Equipment Manufactures Association*). En un principio se expuso en un segundo plano dentro de un pequeño cuarto, ya que Olivetti volcó inicialmente la exposición a la calculadora mecánica LOGOS 27, pero como se observa en la figura 2.2, se hizo una presentación de la Programma 101 dirigida a usuarios no especializados en informática donde se calculaba la órbita de un satélite alrededor de la Tierra —muy apropiado comercialmente, ya que en ese momento se estaba produciendo la conocida carrera espacial—. La demostración tuvo tanto éxito que obligó a los responsables de Olivetti a trasladar la Programma 101 a un punto central de la exposición.

Figura 2.2: Expectación producida en la presentación de la PROGRAMMA 101 en la Exposición Universal de Nueva York (1965)[6].

2.3 Repercusión: prensa y publicidad

El éxito que la PROGRAMMA 101 llegó a tener en la Exposición Universal de Nueva York fue inmediatamente captado por la prensa norteamericana, que rápidamente se hizo eco de la aparición de una computadora de sobremesa. En la figura 2.3 se muestran unos recortes de prensa [26] donde se observa la repercusión que tuvo después de su presentación.

[6]Imagen cortesía de Associazione Archivio Storico Olivetti, Ivrea, Italia. [22]



SAN RAFAEL, CALIF.
INDEPENDENT JOURNAL
D. 36,000

OCT 1 5 1965

COMPUTER ON DESK TOP

An operator inserts a program card into Olivetti Underwood's new desktop computer, the first such unit to be completely self-contained, in a New York City demonstration Thursday. (UPI Telephoto)

Women's Wear Daily
NEW YORK, N. Y.

OCT 1 5 1965

Desktop Computer Introduced by OlivettiUnderwood

NEW YORK. — A compact and economically priced desktop computer that could open the data processing age to small to medium sized firms and stores made a debut here Thursday.

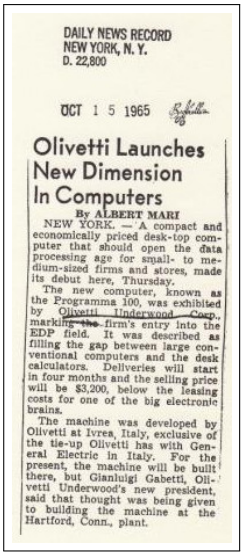
A new desk sized computer known as the Programma 100 was exhibited by Olivetti Underwood Corp. marking the firm's entry into the EDP field. It was described as filling the gap between large conventional computers and the desk calculators. Deliveries will start in four months and the selling price will be $3200, below the leasing costs for one of the big electronic brains.

John Reilley head of the firm's systems division gave a demonstration of the machine doing a difficult task of tracking a satellite's orbit around the earth. He also demonstrated the ease of programming the unit by means of magnetic cards. He added the unit has a large program capacity; can store complex problems inside and outside the unit if needed and can make logical decisions.

The cards carry two programs of 120 instructions each. The Programma computer has 10 registers, each capable of storing 22 digits plus decimal points and arithmetic signs.

It will make its public debut at the Business Machines Mfrs. Exhibition here at the Coliseum, Oct. 25.

Olivetti Underwood will also show at that event an electromechanical calculator, the Logos 27 and a compact electric typewriter.



DAILY NEWS RECORD
NEW YORK, N. Y.
D. 22,800

OCT 1 5 1965

Olivetti Launches New Dimension In Computers

By ALBERT MARI

NEW YORK. — A compact and economically priced desk-top computer that should open the data processing age for small- to medium-sized firms and stores, made its debut here, Thursday.

The new computer, known as the Programma 100, was exhibited by Olivetti Underwood Corp., marking the firm's entry into the EDP field. It was described as filling the gap between large conventional computers and the desk calculators. Deliveries will start in four months and the selling price will be $3,200, below the leasing costs for one of the big electronic brains.

The machine was developed by Olivetti at Ivrea, Italy, exclusive of the tie-up Olivetti has with General Electric in Italy. For the present, the machine will be built there, but Gianluigi Gabetti, Olivetti Underwood's new president, said that thought was being given to building the machine at the Hartford, Conn., plant.

BUSINESS WEEK

October 23, 1965
Fifty cents
A McGraw-Hill publication

'Desk-top' computer is typewriter size

Olivetti Underwood Corp. has developed a desk-top computer that's truly small enough to fit on top of a desk. About the size of an office typewriter, it is classified as a computer because it does such tasks as payroll computation and interest calculations by referring to an internally stored program. Called the Programma 101, it is priced at $3,200.

A payroll clerk would put a magnetic program card in the machine; as many as 120 instructions for such jobs are preprinted on each card. Then the clerk would keypunch in an employee's earnings, and the machine would print out programmed deductions for taxes, medical plans, and the like. **End**

BUSINESS WEEK October 23, 1965

WALL STREET JOURNAL
DALLAS, TEXAS

OCT 1 5 1965

Desk-Top Size Computer Is Being Sold by Olivetti For First Time in U.S.

By a WALL STREET JOURNAL Staff Reporter

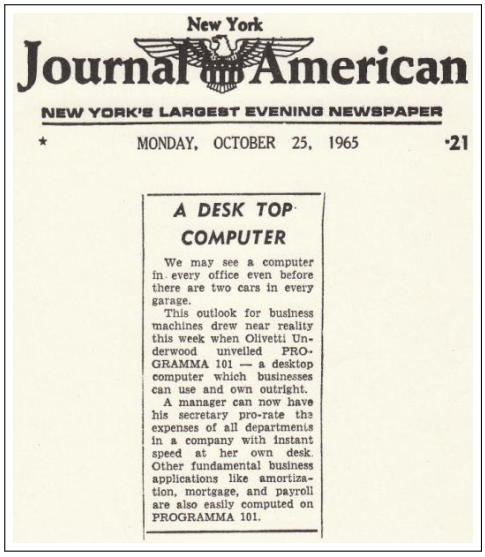
NEW YORK—Olivetti Underwood Corp. is offering its first computer for sale in the U.S.

A small desk-top computer for business and engineering use with sale price of $3,200 was introduced by the company at a press conference.

Olivetti Underwood, a U.S. subsidiary of Ing. C. Olivetti & Co., S.p.A., of Italy, said the computer has the power to make logical decisions. It uses a program card for insertion into the machine to perform operations.

The company said it expects the new computer to fill a gap in the market between desk calculators and large computers.

A calculating machine and an electric typewriter were also introduced.



New York
Journal American
NEW YORK'S LARGEST EVENING NEWSPAPER

★ MONDAY, OCTOBER 25, 1965 •21

A DESK TOP COMPUTER

We may see a computer in every office even before there are two cars in every garage.

This outlook for business machines drew near reality this week when Olivetti Underwood unveiled PROGRAMMA 101 — a desktop computer which businesses can use and own outright.

A manager can now have his secretary pro-rate the expenses of all departments in a company with instant speed at her own desk. Other fundamental business applications like amortization, mortgage, and payroll are also easily computed on PROGRAMMA 101.

November 11, 1965 • ENGINEERING NEWS-RECORD

Keyboard Computer Sits on Desk

The Olivetti Underwood Corp. has introduced a desktop computer that the company says will bridge the gap between desk calculators and full scale computers.

It is a keyboard-operated machine that prints input data, instructions and answers at the rate of 30 characters per second. Programs involving up to 120 instructions may be entered manually through the keyboard, or, if it is a program that is used repeatedly, it may be fed to the machine automatically from a coded magnetic card.

Once programmed, the machine will solve a problem when the variables are entered on the keyboard; no operation instructions are needed, the machine follows the program, printing out the answers automatically.

The machine can solve differential equations, bessel functions and perform numerical integration. Spokesmen for Olivetti Underwood say the machine's program language is extremely simple and can be learned by anyone familiar with mathematical calculations in a very short time.

COMPUTER can take 120 instructions.

In addition, the company is developing and will maintain a library of standard programs in most fields of mathematics, including engineering.

The machine can be bought outright for $3,200 or leased on a monthly basis. OLIVETTI UNDERWOOD CORP., 581 CAPITOL AVE., HARTFORD, CONN.

Figura 2.3: Recortes de prensa norteamericana sobre la PROGRAMMA 101 (1965).

Algunos de los titulares que se pudieron ver a finales de 1965 fueron los siguientes: en el *Independent Journal*, «Computer on desk top»; en el *Women's Wear Daily*, «Desktop Computer Introduced by OlivettiUnderwood»; en el *Daily News Record*, «Olivetti Launches New Dimension In Computers»; en el *Business Week*, «Desk-top computer is typewriter size»; en el *Wall Street Journal*, «Desk-Top Size Computer Is Being Sold by Olivetti For First Time in U.S.»; en el *Journal American*, «A desk top computer» y en el *Engineering News-Record*, «Keyboard Computer Sits on Desk».

La empresa Olivetti aprovechó el momento de popularidad de la Programma 101 y volcó una gran cantidad de recursos para publicitarla por todo el mundo, llegar al público en general y, de esta forma, hacer que entrara en las casas de personas de todo el mundo que no tuvieran conocimientos en informática.

Una muestra de la publicidad lanzada por Olivetti [16] en lengua inglesa son las que se muestran en las figuras 2.4 y 2.5. En estos dos casos los eslóganes dicen: «It has logic, it has a program, it has an electronic memory, it has printing, it is a real computer» y «The Olivetti Programma 101: the do-it-yourself desk-top computer».

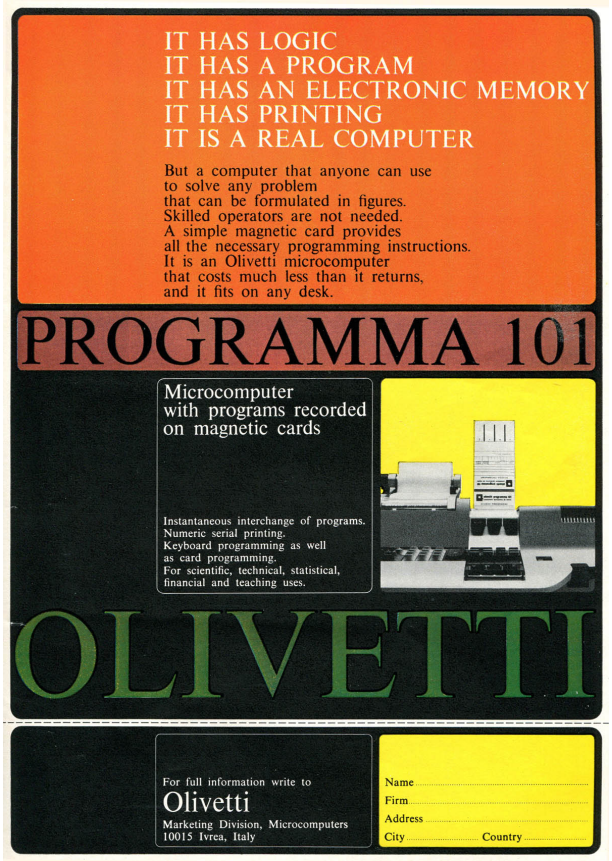


Figura 2.4: Publicidad lanzada por Olivetti para la Programma 101 en lengua inglesa (1965-1966).

Figura 2.5: Publicidad lanzada por Olivetti para la PROGRAMMA 101 en lengua inglesa (1965-1966).

En España, la PROGRAMMA 101 también tuvo un hueco entre la publicidad de aquella época [16] con eslóganes como los que aparecen en las figuras 2.6 y 2.7: «La información pasa a través de Olivetti» y «Un calculador electrónico en la construcción».

Figura 2.6: Publicidad lanzada por Olivetti para la PROGRAMMA 101 en España (1965-1966).

Figura 2.7: Publicidad lanzada por Olivetti para la PROGRAMMA 101 en España (1965-1966).

Como se ha mencionado en la página anterior, Olivetti lanzó una campaña de publicidad de la PROGRAMMA 101 por todo el mundo [16] y una muestra de este despliegue de medios en diferentes países se puede observar en la figura 2.8.

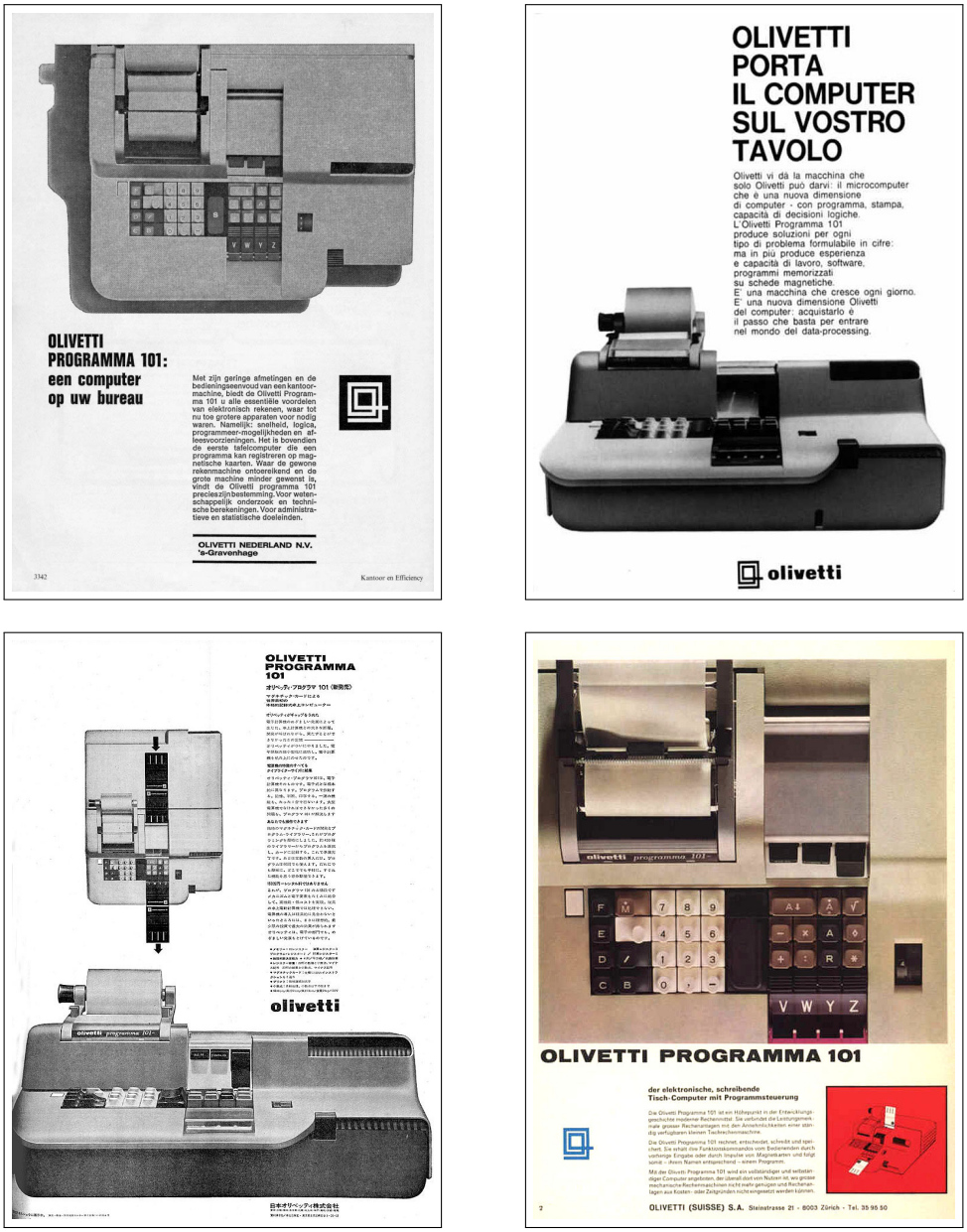


Figura 2.8: De izquierda a derecha y de arriba a abajo: publicidad lanzada por Olivetti para la PROGRAMMA 101 en Holanda, Italia[8], Japón y Suiza[8].

Con esta campaña de publicidad, Olivetti tuvo que realizar una producción masiva de la máquina, por lo que la empresa se vio obligada a crear una nueva línea de producción en Pensilvania (Estados Unidos) llegando a vender 44 000 unidades a un precio de lanzamiento de 3 200 dólares, un precio mucho más reducido que los

[8]Imágenes cortesía de Associazione Archivio Storico Olivetti, Ivrea, Italia. [22]

100 000 dólares que valía la más barata de las grandes computadoras de IBM que en ese momento estaban en el mercado.

La Programma 101 llegó a numerosos ámbitos donde nunca antes se había conocido el uso de ordenadores como es el caso, entre otros, de la medicina, pequeñas empresas, estudiantes, topógrafos y, como gran logro, llegó a introducirse dentro de la vida cotidiana de las personas, algo inimaginable para las grandes computadoras que en ese momento existían.

Otro uso que se le dio a la Programma 101 fue en la planificación de operaciones bélicas en la Guerra de Vietnam (1959-1975) por parte de las fuerzas aéreas norteamericanas, más concretamente para calcular coordenadas en los bombardeos dirigidos desde tierra del *B-52 Stratofortress*.

La agencia espacial norteamericana NASA compró en 1966 más de 10 unidades que utilizó en la elección de puntos de aterrizaje y en el cálculo de las maniobras de alunizaje [22] de la misión Apolo 11 que, en julio de 1969, llevó al hombre a la luna por primera vez. Como declaró en 2006 David W. Whittle, miembro de la NASA [17]:

> En el momento de Apolo 11 teníamos una computadora de escritorio, o algo así, o se parecía, llamada Olivetti Programma 101. Era una especie de supercalculadora. Era probablemente un cuadrado de pie y medio, y cerca de 8 pulgadas de altura. Podía sumar, restar, multiplicar y dividir, pero podía recordar una secuencia de estas operaciones y guardaba esa secuencia en una tarjeta magnética, una cinta magnética que era cerca de un pie y medio de largo y dos pulgadas de ancho. Así que podías escribir una secuencia, una secuencia de programación y cargarla ahí — La antena de alta ganancia del módulo lunar no era muy inteligente, no sabía dónde estaba la tierra. [...] Teníamos que correr cuatro programas separados en la Programma 101 [...]

De la participación de la Programma 101 en este hito histórico también hay documentación gráfica que lo demuestra como las fotografías que se pueden ver en la figura 2.9[9].

[9]Imágenes cortesía de Associazione Archivio Storico Olivetti, Ivrea, Italia. [22]

Figura 2.9: Fotografías tomadas en agosto de 1969 junto a personal de la NASA.

2.4 Situación empresarial

Como se observa en la tabla 2.1, en los años 50 del siglo XX, Estados Unidos lideraba el mundo de la informática con nueve empresas, mientras tanto, una empresa situada en Ivrea (Italia) especializada en máquinas de escribir y calculadoras mecánicas llamada Olivetti y presidida por Adriano Olivetti invertía en tecnologías electrónicas creando el laboratorio de I+D en informática más importante de Europa.

Revenues of selected computer and electronic companies, 1955

Company	Annual sales	Net profit	Employees
GE	$2.96 billion	$213 M	210,000
Western Electric*	$1.5 billion	$55 M	98,000
RCA	$940 M	$40 M	70,500
IBM	$461 M	$46.5 M	46,500
NCR	$259 M	$12.7 M	37,000
Honeywell	$229 M	$15.3 M	25,000
Remington Rand**	$225 M	$12.2 M	37,000
Raytheon	$177 M	$3.5 M	18,700
Burroughs	$169 M	$7.8 M	20,000

Source: Data from *Fortune* (July 1955).
* Western Electric was the manufacturing arm of AT&T, which owned and controlled it. AT&T's total revenues for 1955 were greater than GE's, RCA's, and IBM's combined.
** In 1955 Remington Rand merged with the Sperry Corporation, a company with $441 million in sales, mostly defense-related.

Tabla 2.1: Las nueve principales compañías del sector de la informática y electrónica en 1955 [24].

Tras la muerte de Adriano Olivetti en 1960, la empresa es heredada por su hijo Roberto Olivetti que dos años más tarde (1962) será quien mande desarrollar una calculadora electrónica y que llegaría a convertirse en una computadora diferente, la Programma 101.

En 1964, poco antes de que la Programma 101 se diera a conocer, la empresa Olivetti se vio afectada por una crisis económica que obligó, en mayo de ese mismo año, a la entrada de nuevos accionistas (Pirelli, Mediobanca, IMI, la Centrale y Fiat) que decidieron desinvertir en electrónica y vender el 75 % de la sección electrónica a la multinacional americana General Electric junto con todos los proyectos que allí se estaban desarrollando. Sin embargo, para evitar que la Programma 101 fuera a parar a manos de la empresa General Electric, Perotto y los miembros de su equipo cambiaron en todo el proyecto la denominación de «computadora» por

«calculadora», de esta manera Perotto y su equipo de desarrollo se ocultaron dentro de la estructura corporativa de Olivetti y pudieron terminar su trabajo libremente.

Después de la presentación de la Programma 101 en la Exposición Universal de Nueva York y su posterior éxito, los desarrolladores pensaron que debían proteger su producto frente a sus competidores, ya que sus partes podrían ser analizadas y copiadas, tal y como se pudo comprobar en 1968. Por ello, en marzo de 1965 presentaron en Estados Unidos la patente número 3.495.222 que se puede ver en el apéndice B [28], la cual está firmada por Pier Giorgio Perotto y Giovanni De Sandre. Inmediatamente después, Olivetti les hizo firmar una declaración de cesión a la empresa de todos los derechos a cambio de un dólar, que decía [27] según el propio Perotto en 1995.

> Cedo a la Olivetti, por un dólar [...], los derechos consiguientes a la invención descrita en la instancia de la patente número 3.495.222, depositada el 1 de marzo de 1965, a nombre de P.G. Perotto y otros, de título: *Program Controlled Electronic Computer*.

Entre las miles de unidades de la Programma 101 vendidas por Olivetti, 100 fueron compradas por la empresa Hewlett Packard y más tarde —en 1968— apareció en el mercado la calculadora HP 9100. Esta calculadora tenía las mismas características que la de Olivetti pero con capacidades ampliadas, incluso utilizaba tarjetas magnéticas. Debido a las similitudes que existían entre estas dos máquinas, Hewlett Packard pagó, sin ir a juicio, 900 000 dólares a la empresa italiana en concepto de derechos o regalías y poder así utilizar la patente de la Programma 101 que años antes Pier Giorgio Perotto y Giovanni De Sandre habían cedido por la cifra simbólica de un dólar.

Desde 1968, la competencia fue capaz de ponerse al día introduciendo en el mercado productos capaces de competir con la Programma 101, deshaciendo la oportunidad de obtener una posición de liderazgo en el mercado de la informática que, a partir de los años 70, culminó con los ordenadores personales —IBM en 1981— sin que Olivetti reaccionara al respecto.

Además de la sospecha de un complot internacional contra los intereses de Italia, Pier Giorgio Perotto en el capítulo 6 «Un'occasione perduta?» de su libro [27], culpa a la dirección de la empresa de la ocasión perdida sobre los mercados, ya que en su opinión no protegió ni promovió la transferencia de tecnología (*know-how*) que tenía la sección electrónica de Olivetti en el momento de la venta a la General Electric. La directiva tenía una visión arcaica del poder donde se limitaban a invertir en tiempos de bonanza y desinvertir en periodos de crisis sin importarles nada todo aquello que estuviera fuera de su ámbito empresarial, como la cultura humanística y técnica. Esta forma de actuar de la dirección respecto a la Programma 101 fue más tarde objeto de estudio por la Universidad de Harvard como una ocasión perdida de negocio.

Tras el éxito de la Programma 101, la empresa Olivetti no pudo remontar al no poder seguir el ritmo de la evolución en el desarrollo de arquitecturas en el

campo de la informática, incluso a pesar de los intentos realizados con una gama de productos sucesores de la Programma 101 —P101— donde se adaptaban a los estándares de facto que iban apareciendo en el mercado como es el caso de: la P102, con puerto serie para conectarle periféricos; la P203, con memoria ampliada y conexión a una máquina de escribir eléctrica —Tekne3—; la P602, con mayor memoria, ROM con funciones predefinidas, juego de instrucciones ampliado y con una gama de periféricos; la P603, como la P602 pero conectada a la Tekne3, como la P203.

CAPÍTULO 3

Arquitectura interna

En este capítulo se describen detalladamente los aspectos más novedosos de la arquitectura interna de la PROGRAMMA 101. Entre estos aspectos se encuentran el tipo de electrónica empleada en su construcción, el tipo —tecnología— de memoria utilizada, las tarjetas magnéticas que se usaban por primera vez, su diseño —estética— industrial y, finalmente, se fundamenta la facilidad de uso de este dispositivo.

Pero antes y como se comprueba en las figuras 3.1 y 3.2, se ha de remarcar que la arquitectura de esta máquina sigue las pautas marcadas por John von Neumann (1903-1957), en el modelo que lleva su nombre y que fue publicado el 30 de junio de 1945 por la Universidad de Pensilvania y la armada de los Estados Unidos en un borrador llamado «*First Draft of a report to the EDVAC*» [8].

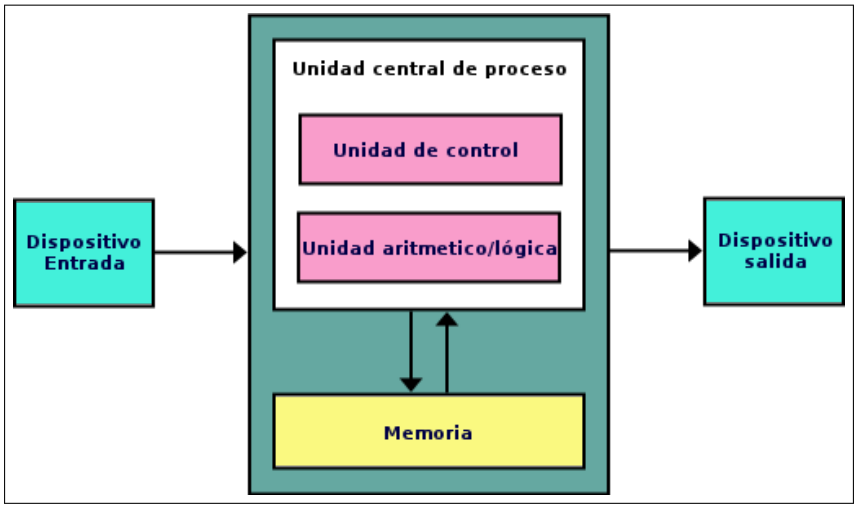


Figura 3.1: Arquitectura de von Neumann

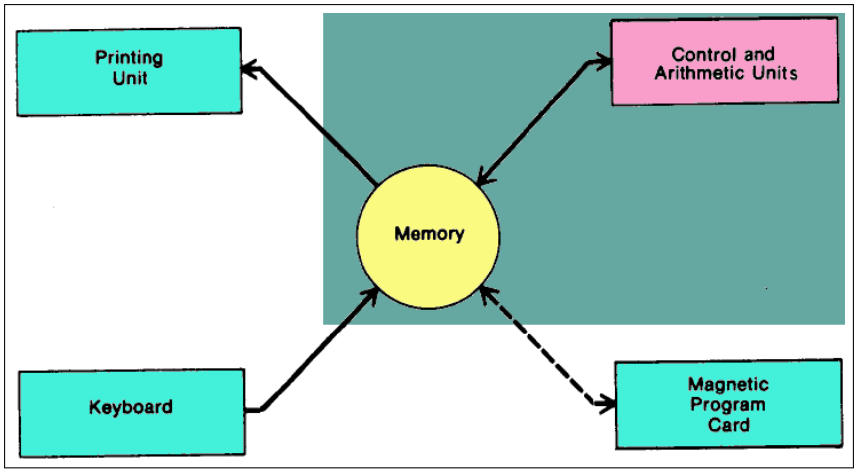


Figura 3.2: Arquitectura de la Programma 101 [21].

Este modelo describe una arquitectura compuesta por una unidad de control conectada a una memoria y estas dos a su vez se comunican con el usuario a través de los dispositivos de entrada y salida, que en el caso de la Olivetti está formada por el teclado y la impresora respectivamente; además, esta tiene la unidad de lectura/escritura de tarjetas magnéticas que cumple con ambas funciones a la vez.

3.1 Tipo de electrónica y memoria

Como se ha mencionado en el apartado 2.1, la Programma 101 fue desarrollada durante la segunda generación de la historia de las computadoras, por lo que estaba construida con elementos electrónicos discretos como los transistores —fueron utilizados alrededor de 670 del modelo 2N708—, los diodos, las resistencias y los condensadores. En las figuras 3.3 y 3.4 se muestran el transistor que hizo posible el reducido tamaño de la Programma 101 y el interior de esta, dejando a la vista sus componentes electrónicos, respectivamente.

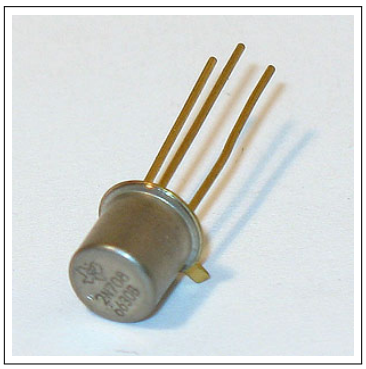


Figura 3.3: Transistor 2N708 NPN con encapsulado metálico TO-18 utilizado en la construcción de la Programma 101.

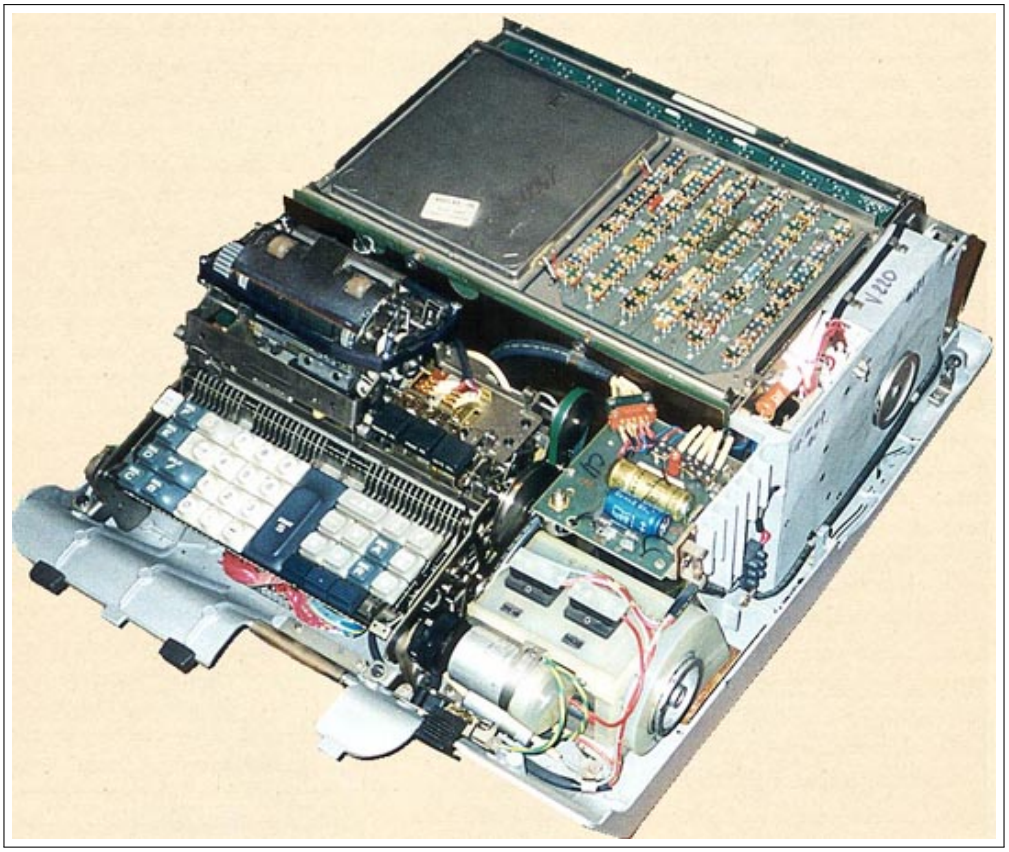


Figura 3.4: Vista del interior de la Programma 101.

Para entonces aún no existían los circuitos integrados típicos de la tercera generación, y los componentes mencionados debían ser suficientes para construir esta máquina. Ante este reto, los desarrolladores idearon una memoria de línea de retardo acústica magnetostrictiva con filamento de metal, un tipo de memoria que se debe actualizar (o refrescar) y es secuencial, al contrario de las memorias actuales, que son aleatorias.

Este tipo de memoria consiste principalmente en un filamento metálico enrollado para alcanzar una amplia longitud —6,5 metros en el caso de la Programma 101— en el menor espacio posible. Conectado a este alambre se encuentran unos transceptores formados por material piezoeléctrico que convierten señales eléctricas en ondas acústicas y viceversa.

Como se representa en la figura 3.5, el funcionamiento de este tipo de memoria se basa fundamentalmente en el retardo que le supone a una onda acústica en transmitirse a lo largo de un medio, así pues, los datos se introducen en dicho medio a través de un transductor y, tras un retardo —2,2 milisegundos en la Programma 101—, esta señal se encuentra al final del alambre recogida por el otro transductor y retroalimentada formando un bucle cerrado y así mantiene, refresca o memoriza la información introducida.

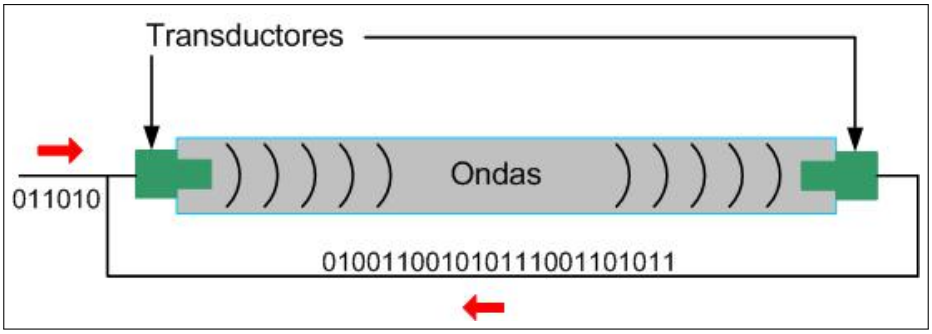


Figura 3.5: Fundamento de la memoria de línea de retardo[1].

En la primavera de 1962 y basándose en esta tecnología, se construyó el prototipo de memoria que se observa en la figura 3.6, compuesto por un filamento metálico de una longitud de 1 280 metros, dos bobinas —color rojo— como transductores para la escritura y una bobina —color amarillo— como transductor para la lectura. Este prototipo tenía una capacidad de 256 bits y un retardo de 256 microsegundos.

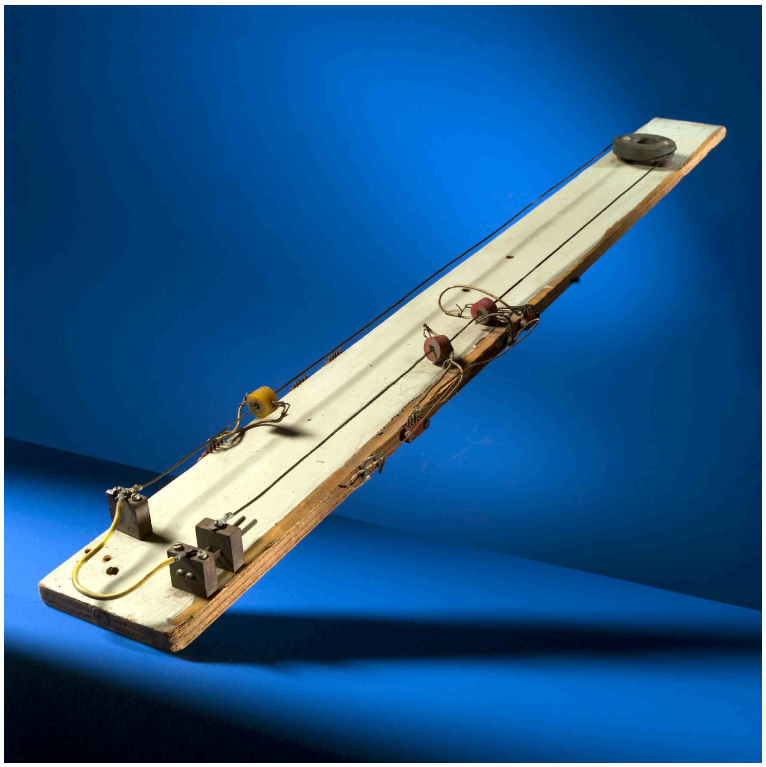


Figura 3.6: Prototipo de memoria construida para el desarrollo de la Programma 101.[2]

[1] `http://elmecanismodeanticitera.blogspot.com.es/2011/07/lineas-de-retardo.html`
[2] Imagen cortesía de Associazione Archivio Storico Olivetti, Ivrea, Italia. [22]

Como se puede observar en la figura 3.7, la unidad de memoria de la Programma 101 construida con la tecnología descrita en el párrafo anterior tenía un tamaño de $18 \times 19 \times 2$ cm, un tamaño muy reducido en comparación a las memorias de los grandes computadores existentes en esa época. En la Programma 101, la memoria tenía una capacidad para almacenar un total de 240 caracteres en codificación BCD donde cada dígito decimal es codificado con 4 bits, por tanto la capacidad de memoria total era de 960 bits (240×4).

Figura 3.7: Unidad de memoria de la Programma 101 [16] formada mediante una línea de retardo acústica magnetostrictiva con filamento metálico.

3.2 Tarjeta magnética

Las tarjetas magnéticas eran otra de las novedades que incorporaba la Programma 101 —figuras 3.8 y 3.9—. Hasta el momento solo se disponía de grandes y pesados rollos de cinta magnética, pero estos no eran un medio factible para la nueva máquina que se estaba ideando, por lo que, cuando estaban desarrollando la Programma 101 diseñaron también dichas tarjetas magnéticas, su correspondiente dispositivo de lectura/escritura e incluso, un lenguaje de programación sencillo para que personas no especializadas pudieran utilizarlas. Estas tarjetas magnéticas eran mucho más manejables debido a su reducido tamaño, se podían almacenar programas y datos, así como ser transportadas fácilmente para su posterior uso.

A partir de la aparición de las tarjetas magnéticas de la Programma 101, otros fabricantes de computadores, como por ejemplo IBM, copiaron la idea y fue evolucionando hasta llegar a desarrollar los disquetes y todos los medios magnéticos que se utilizan en la actualidad.

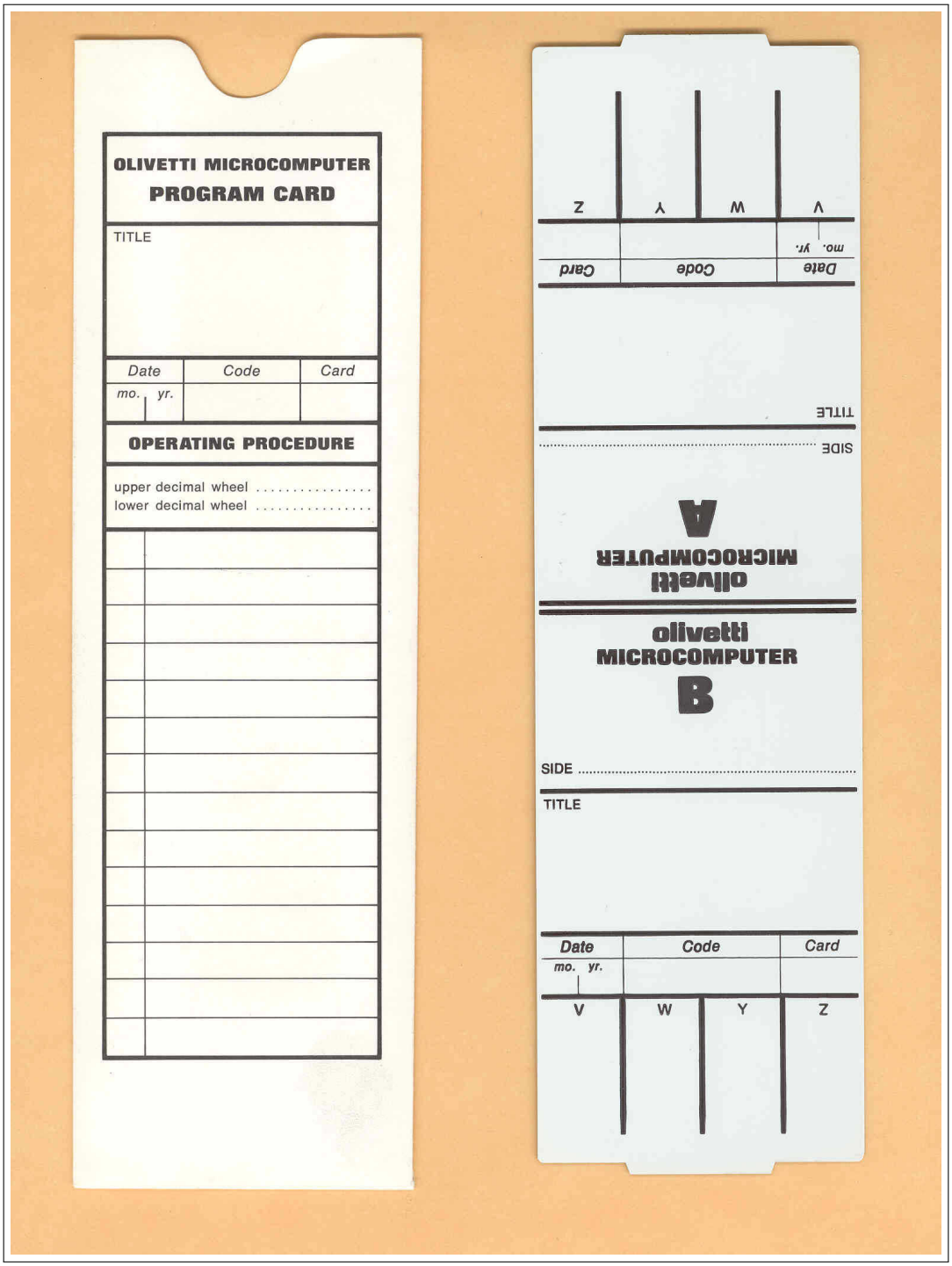


Figura 3.8: Tarjeta magnética usada por la Programma 101 [16].

Figura 3.9: Modo de utilización de la tarjeta magnética [16].

3.3 Estética

Otro punto novedoso de esta máquina está relacionado con sus dimensiones y aspecto, ya que hasta la fecha, todas las computadoras eran muy grandes y en ningún caso tenían una estética agradable. El equipo de desarrolladores consiguió construir la Programma 101 con un tamaño de $48 \times 61 \times 19$ cm y un peso de 35,5 kilogramos.

La estética y ergonomía, que nunca antes se habían tenido en cuenta en el diseño de este tipo de máquinas, fue encargada personalmente por el presidente de la empresa Roberto Olivetti al arquitecto y diseñador italiano Mario Bellini. Los diseñadores buscaban una máquina que se convirtiera en un objeto personal, que interaccionara con los usuarios y fuera fácil su comprensión. Tal como mencionó el diseñador en «Programma 101 — memory of the future» (2011) [17].

> Recuerdo que un día recibí una llamada de Roberto Olivetti: «quiero verte para un proyecto complejo que estoy construyendo». Se trataba del diseño, no de una caja conteniendo mecanismos y circuitos impresos, sino un objeto personal, algo que tenía que vivir con una persona, una persona con su silla sentada ante una mesa o escritorio y que tenía que empezar una relación de comprensión, de interacción, algo muy nuevo, porque antes de esa fecha las computadoras eran tan grandes como un armario. Con un armario nosotros no tenemos ninguna relación: de hecho, los armarios más bellos desaparecen en la pared. Pero esto no era un armario o una caja, se trataba de una máquina diseñada para ser parte de su séquito personal.

Bellini tuvo que integrar en el diseño de la Programma 101 las partes correspondientes a la interfaz de usuario, tanto para la entrada como para la salida de datos. La interfaz de entrada consistía en un teclado y una rueda de decimales para indicar la precisión de las operaciones aritméticas, mientras que la interfaz de salida, consistía en un *display* formado por dos lámparas —una verde para las confirmaciones y otra roja para los errores— y en una impresora. Dicha impresora muestra información de salida sobre un rollo de papel con una velocidad de hasta 30 caracteres por segundo [22].

Los componentes mencionados se pueden observar con más detalle en las figuras 3.10, 3.11 y 3.12 junto al diseño global que se muestra en las figuras 3.13 y 3.14 [16], el diseño de la Programma 101 fue tan novedoso que fue expuesta en el Museo de Arte Moderno de Nueva York [27].

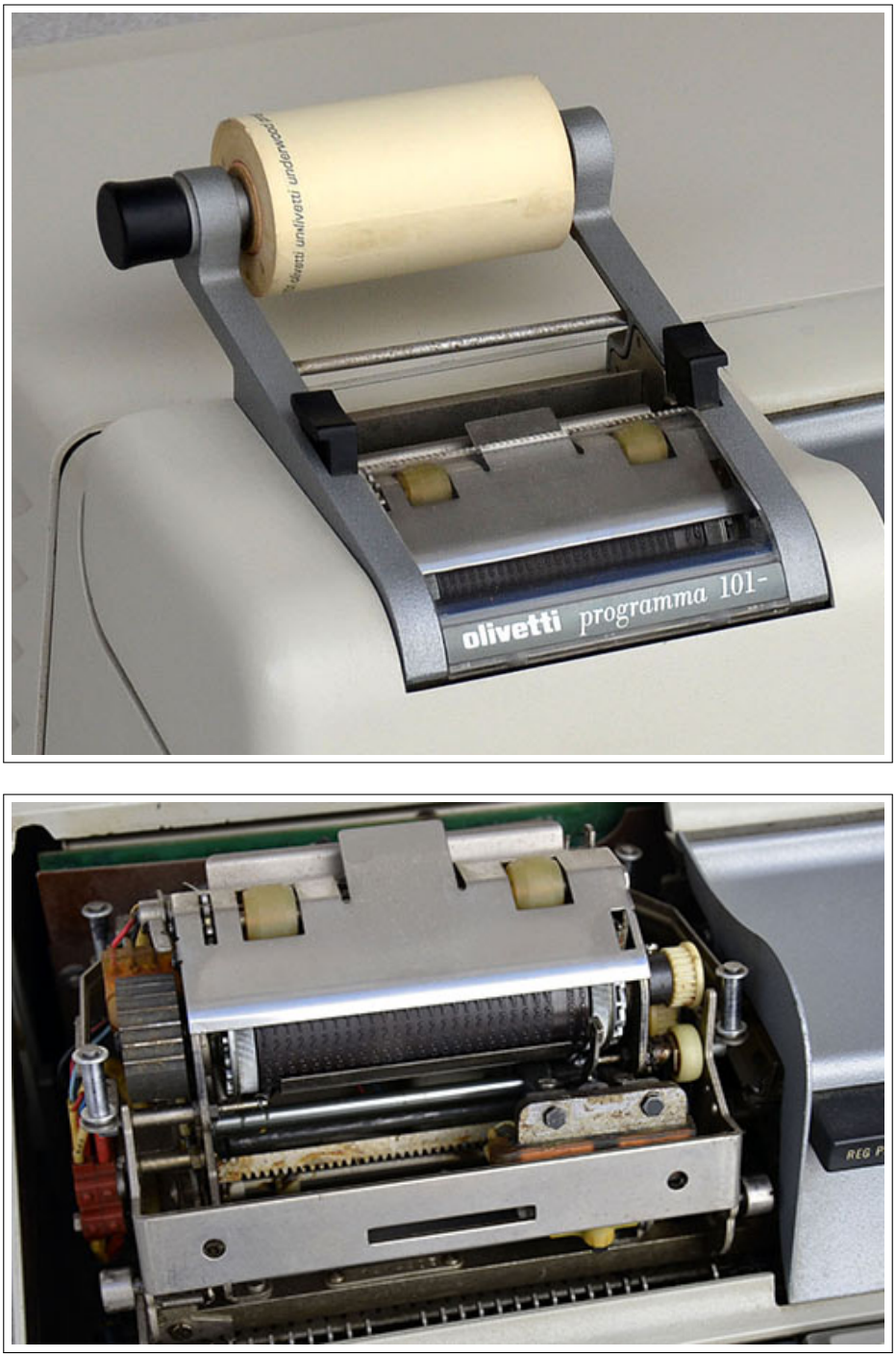


Figura 3.10: Diseño e interior de la impresora de la Programma 101.

Figura 3.11: Diseño del teclado y del lector/grabador de tarjetas magnéticas de la Programma 101.

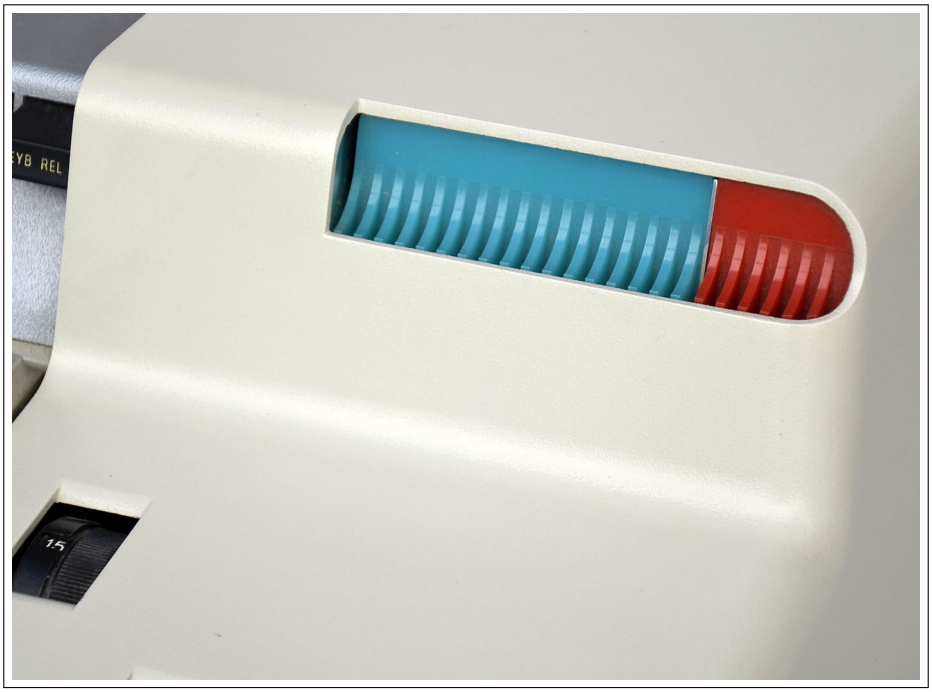


Figura 3.12: Diseño del *display* de la Programma 101.

Figura 3.13: Perspectiva delantera del diseño realizado por Mario Bellini.

Figura 3.14: Perspectiva trasera del diseño realizado por Mario Bellini.

El diseño de la PROGRAMMA 101 estuvo orientado a que el usuario que manejara esta computadora no tuviera grandes conocimientos en informática y llevar su uso a todos los ámbitos posibles. Para ello se creó un lenguaje de programación fácil de usar y que con poca práctica y poco tiempo de aprendizaje se pudiera dominar. En este aspecto, Olivetti cambió la filosofía que había hasta entonces, en la que el perfil de los usuarios de las computadoras existentes era muy especializado y con amplios conocimientos en la máquina que se utiliza, usándose solo en ámbitos muy restringidos.

CAPÍTULO 4

Programación

Por un lado, en este capítulo se van a describir tanto el lenguaje de programación de la Programma 101, como algunos aspectos de la gestión de memoria, ya que todo ello será necesario para comprender cómo funciona esta máquina. Por otro lado, se va a realizar una comparación con el procesador MIPS R2000, el cual se estudia en asignaturas de la titulación del Grado en Ingeniería Informática relacionadas con la estructura y la arquitectura de computadores. Con esta comparativa se pretende tomar una referencia y así comprender mejor las características de la Programma 101.

4.1 Los registros y la memoria

Para entender el funcionamiento de la Programma 101 es necesario saber cómo está organizada su memoria. Como se indicó en el capítulo anterior, tiene una capacidad de 240 caracteres, divididos en 10 registros de 24 caracteres o dígitos, gracias a los cuales la Programma 101 tenía una capacidad de cómputo muy elevada para la época en la que apareció.

De estos diez registros, dos están reservados para el programa, teniendo entre los dos una capacidad para 48 instrucciones, destinándose el resto de registros —denominados M, A, R, B, C, D, E y F— al almacenamiento de datos repartiendo su capacidad en 22 dígitos para el dato en sí, otro para indicar la posición del punto decimal y el último para el signo. Entre estos ocho registros había usos especiales que se describen a continuación.

Hay tres registros —M, A y R— que tienen un papel especial en el funcionamiento de la Programma 101; el M es el encargado de guardar las entradas que proporciona el teclado y distribuir esa información al registro adecuado; el A hace la función de acumulador, en él se guardan los resultados de las operaciones realizadas y las retiene; y en el R nos encontramos con el resultado de las operaciones

aritméticas que se hayan realizado actualizándose tras la finalización de cada una de ellas.

De los otros cinco registros, tres de ellos —D, E y F— pueden almacenar instrucciones en caso de que el código de programación sea muy largo sumándolos con los dos registros destinados a instrucciones, de esta forma obtenemos un total de 120 instrucciones —número máximo que puede manejar—.

Tanto estos tres como los dos registros restantes destinados exclusivamente a almacenar datos, pueden ser divididos en dos, encontrándonos en una situación en la que podremos guardar once datos con punto decimal y signo. En el caso de dividir los registros, estos son referenciados como b, B, c, C, d, D, e, E, f y F. Esta división se realiza mediante el comando "/" de tal forma que B/ = b. En la figura 4.1 se puede contemplar las posibles combinaciones que nos permiten estos registros.

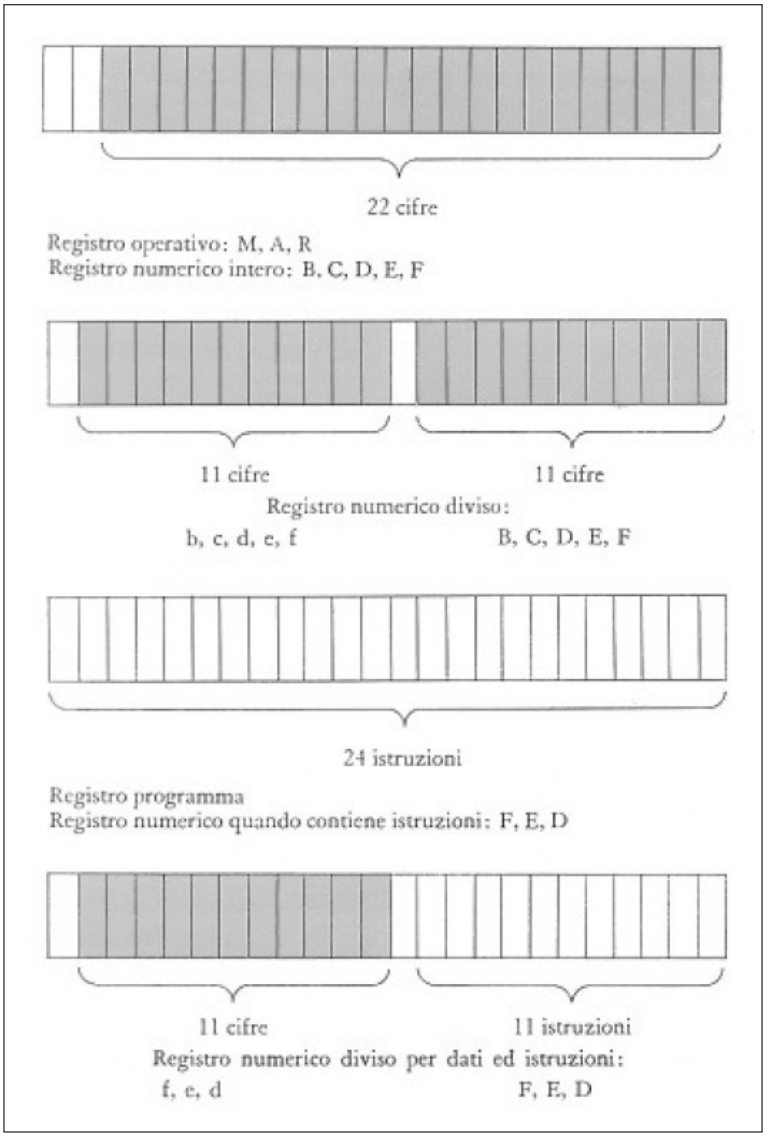


Figura 4.1: Posibles configuraciones de los registros. [20]

4.2 Instrucciones

Ahora que se conoce la estructura y organización de los registros de la PROGRAMMA 101, se procede a describir las 16 instrucciones que se pueden realizar en este computador clasificadas por sus respectivos propósitos:

- **Instrucciones generales:**
  - **"S" (Stop/Start):** Cuando un programa se está ejecutando y se encuentra con esta instrucción, la PROGRAMMA 101 espera a que se le introduzca por teclado un valor, posteriormente hay que pulsar "S" para validar el dato y reanudar la ejecución.
  - **"*" (Clear):** Borra el contenido del registro especificado (por ejemplo B*).
  - **"◇" (Print):** Imprime el contenido del registro especificado con la cantidad de decimales que se especifique en la rueda de decimales. Si en lugar de especificar un registro, aparece en el código "/◇" realiza una impresión de una línea en blanco.
- **Transferencia de datos:**
  - **"↓" (To A):** Pasa el contenido del registro especificado al acumulador, si no se especifica este se asume que es M (por ejemplo ↓: M → A).
  - **"↑" (From M):** Traslada el contenido de M al registro especificado (por ejemplo B/↑: M → b) menos en dos casos, R↑ que no es operativo y A↑ que se usa en la introducción de constantes como instrucciones [20, 19].
  - **"↕" (Exchange):** Intercambia el contenido del registro especificado y el acumulador (por ejemplo B↕: B ⇄ A) si no se indica el registro se asume que es M, además hay dos casos especiales, A↕ = |A| y R↕ = R → A.
  - **"RS" (D-R Exchange):** Intercambia el contenido de los registros d y D (de ambos) con el contenido del registro R. Esta instrucción tiene dos usos muy especiales: por un lado, y más importante, la carga de un programa desde varias tarjetas magnéticas. Para ello, temporalmente se almacenan d y D en R hasta que se ha leído la siguiente tarjeta, ya que en ese lapso de tiempo los datos que almacena R no son modificados. Por otro lado, y como segundo uso, borrar D cuando este almacena una instrucción mediante la secuencia de estas dos instrucciones: R*, RS.
  - **"/↕" (Decimal To M):** Pasa la parte decimal del contenido del registro A al registro M.
- **Operaciones aritméticas:** Todas las operaciones de este tipo se realizan en dos fases; la primera traslada el contenido del registro que se especifica a M, si no se indica ninguno, se usa el propio registro M; en la segunda fase se realiza la operación de este con el registro A, el resultado es guardado en R y en A, pero en este último se trunca según el número de decimales que se ha configurado en la rueda destinada a dicha función.

- "+" **(Addition):** Suma el contenido del registro especificado con el de A.
- "−" **(Subtraction):** Resta al contenido de A el valor guardado en el registro especificado.
- "×" **(Multiplication):** Multiplica los valores de los registros A y el especificado.
- "÷" **(Division):** Divide el contenido de A por el contenido del registro especificado; en este caso el cociente es guardado en A y el resto en R.
- "$\sqrt{\ }$" **(Square Root):** Realiza la raíz cuadrada del contenido del registro especificado y guarda el resultado truncado en A; en esta operación R no se usa.
- "A↕" **(Absolute Value):** Como se indicó en el apartado de operaciones de transferencia de datos A↕ = |A|.

• **Operaciones de salto:** Estas operaciones son necesarias para realizar bucles, seleccionar subrutinas indicadas por el usuario o condicionarlas al valor del registro A. Para llevar a cabo un salto hace falta un punto de origen y una referencia a este —destino—, cuando el hilo de ejecución se encuentra con un punto de origen salta hasta su respectiva referencia.

- **Saltos incondicionales:** Salta al encontrarse con un punto de origen —△ = V, W, Y o Z, estos son los más habituales y están incluidos en el teclado para su selección por parte del usuario— hasta su respectiva referencia, en la tabla 4.1 se indican todas las posibles combinaciones.

PUNTO DE ORIGEN	REFERENCIA A PUNTO
△	A△
C△	B△
D△	E△
R△	F△

Tabla 4.1: Relación entre orígenes y destinos en los saltos incondicionales.

- **Saltos condicionales:** Este tipo de salto está condicionado al valor que contiene el registro A; si este es mayor que 0 se produce el salto, en caso contrario sigue con la siguiente instrucción. Este comportamiento es muy útil para la realización de bucles condicionados en su finalización tal y como se podrá ver en el capítulo siguiente. En este caso la tabla 4.2 muestra los puntos de origen y sus respectivas referencias.

PUNTO DE ORIGEN	REFERENCIA A PUNTO
/△	A/△ = a△
C/△ = c△	B/△ = b△
D/△ = d△	E/△ = e△
R/△ = r△	F/△ = f△

Tabla 4.2: Relación entre orígenes y destinos en los saltos condicionales.

4.3 Introducción de constantes

Durante la programación de la PROGRAMMA 101 se hace necesario la utilización de constantes, como por ejemplo el número pi (π = 3,14159...) para cálculos de trigonometría. Por lo tanto, estos valores hay que introducirlos y guardarlos en algún registro a elegir entre el D, E, F o alguna de sus divisiones (d, e y f), es decir, se podrían guardar hasta tres constantes de 22 dígitos o seis de 11 cada una.

El proceso de introducción de dichas constantes es muy simple: estando en una situación donde los botones *RECORD* y *PRINT* están en *OFF*, se teclea el valor de la constante y a continuación se le indica el registro seguido de la tecla ↑ (por ejemplo E↑), de esta forma se introduce el valor tecleado en el registro indicado.

Otra situación se produce cuando tanto el programa como las constantes están en una tarjeta magnética. Como esta solo almacena instrucciones, lo que se necesita es una secuencia de estas para generar las constantes en tiempo de ejecución, es decir, constantes como si fueran instrucciones. Para realizar estas operaciones se parte de una situación donde el botón *PRINT* esta en *ON* y el botón *RECORD* en *OFF*, se pulsa *RESET* y se inicia la introducción de instrucciones con "A/↑", dichas instrucciones están formadas por una parte izquierda y otra derecha que, combinándolas, adquirirán nuevos significados según las tablas 4.3 y 4.4.

IZQUIERDA	SIGNIFICADO
R	Dígito positivo.
D	Dígito positivo de mayor orden.
F	Dígito negativo.
E	Dígito negativo de mayor orden.
/	Punto decimal (incluye un entero).

Tabla 4.3: Parte izquierda de las instrucciones de introducción de constantes.

DERECHA	SIGNIFICADO
S	0
↓	1
↑	2
↕	3
+	4
-	5
x	6
÷	7
◇	8
*	9

Tabla 4.4: Parte derecha de las instrucciones de introducción de constantes.

Cuando se ha finalizado la introducción de todas las constantes con este método se ha de almacenar a continuación la instrucción "A↑", así pues, para almacenar en la tarjeta magnética el valor −19,62 se ha de guardar la secuencia de instrucciones que se observa en la tabla 4.5.

A/↑	Inicio de la secuencia de instrucciones.
R↑	2
Rx	6
R/*	9.
E↓	−1 y fin de la constante.

Tabla 4.5: Ejemplo de cómo se introduce una constante.

4.4 Comparativa con el MIPS R2000

El procesador MIPS R2000 —enero de 1986— es considerado didácticamente como un punto de referencia para las asignaturas relacionadas con la arquitectura de computadores que son impartidas en la titulación de Grado en Ingeniería Informática. Es por ello que en este apartado se va a realizar una comparación entre este y la Programma 101, de esta forma nos podremos hacer una idea más precisa del potencial que tenía esta última teniendo en cuenta su contexto histórico. El procesador MIPS R2000 se muestra en la figura 4.2.

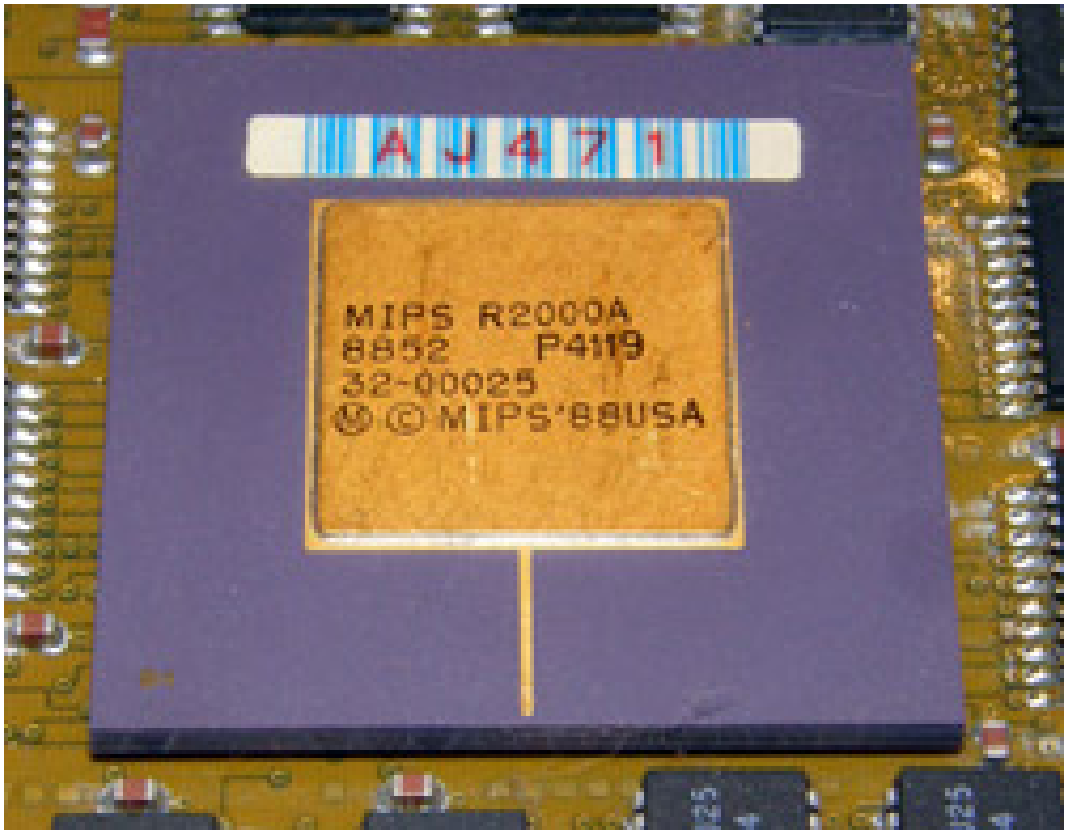


Figura 4.2: Procesador MIPS R2000[1].

Respecto a la memoria, el MIPS R2000 dispone de 32 registros de 32 bits cada uno, bastante más de la memoria que disponía la Programma 101 —10 registros de 96 bits—, pero además, el R2000 tiene dos registros de 32 bits dedicados expresamente al cálculo de multiplicaciones y divisiones enteras y otro más para el contador de programa. Como se puede deducir fácilmente, el R2000 tiene mucha más capacidad de memoria que la Programma 101 con sus 960 bits. Además, el R2000 puede direccionar memoria RAM (*Random-Access Memory*) con un ancho de bus de 32 bits y pudiendo acceder hasta 2^{30} posiciones de memoria ($32 \times 2^{30} = 32$ Gib $= 4$ GiB)[2], algo inexistente en la Programma 101.

El MIPS R2000 por sí solo puede realizar operaciones enteras, para poder operar con números reales, debe ser ayudado por una FPU (*Float Point Unit*) o coprocesador llamado MIPS R2010 —vease figura 4.3—. Gracias a esta unidad de coma flotante, el R2000 dispone de 32 registros más, cada uno de 32 bits, e incrementa su potencial de cálculo ampliando su juego de instrucciones.

[1] `http://www.xanthos.se/~joachim/collection.html`
[2] ISO - International Organization for Standardization. ISO/IEC 80000-13:2008

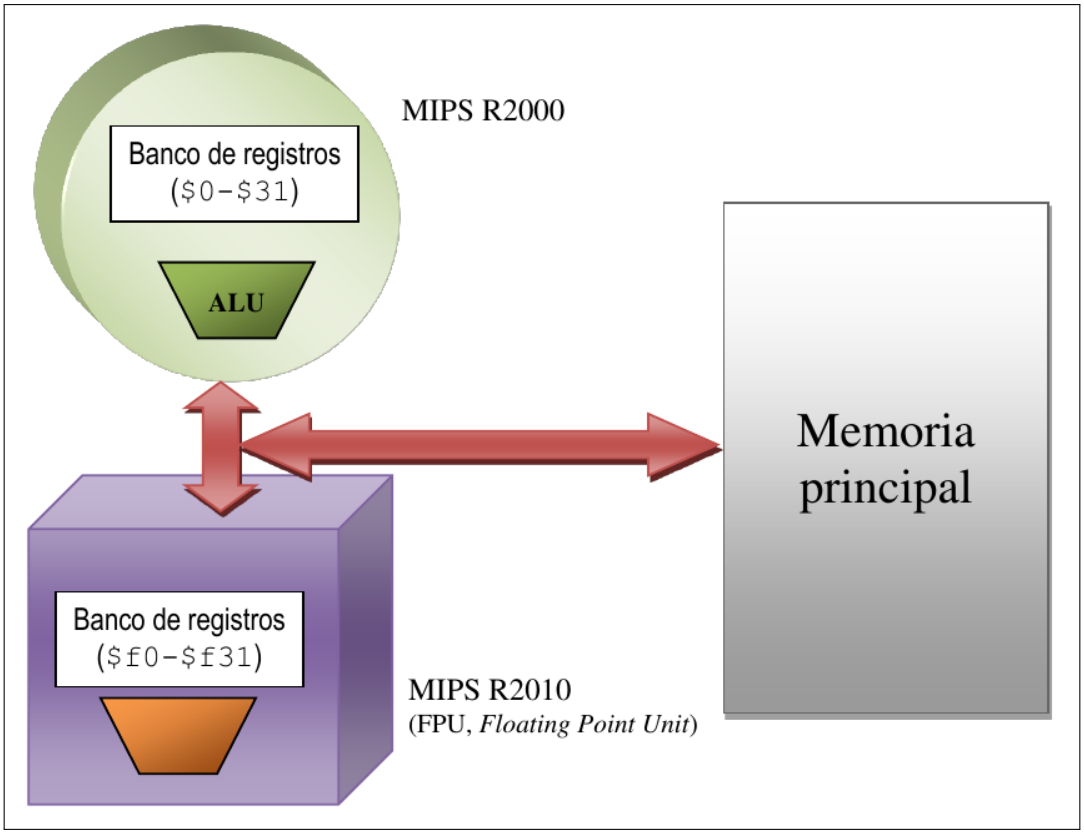


Figura 4.3: Diagrama del procesador MIPS R2000 junto al coprocesador MIPS R2010.

Debido a la gran diferencia existente en la arquitectura del MIPS R2000 —sin coprocesador— respecto a la PROGRAMMA 101, y por tanto un potencial más alto del primero respecto de la segunda, el R2000 dispone de un lenguaje ensamblador mucho más amplio, de hecho, muchas de las instrucciones del MIPS R2000 no tienen sentido en la PROGRAMMA 101, como son las correspondientes a la carga —`lw`, `lb`, `li`, ...— y almacenamiento —`sw`, `sb` y `sh`— en memoria, instrucciones lógicas —`and`, `andi`, `or` y `ori`—, de desplazamiento —`sll`, `srl` y `sra`— y comparaciones —`slt`, `seq`, `sne`, ...—.

Además, el lenguaje ensamblador del MIPS R2000 dispone de más variedad de instrucciones a la hora de realizar las operaciones aritméticas, ya que puede llevarlas a cabo distinguiendo entre operaciones con o sin signo y si es inmediata o no. Sin embargo, las operaciones de la PROGRAMMA 101 para calcular la raíz cuadrada y el valor absoluto de números enteros no existen en el R2000, a no ser que trabaje con números reales junto al R2010. En la tabla 4.6 se muestra una comparativa entre ambas.

PROGRAMMA 101	MIPS R2000
$+$	`add`, `addu`, `addi`, `addiu`
-	`sub`, `subu`
$\times$	`mult`
$\div$	`div`
$\sqrt{\ }$	——
$A\updownarrow \equiv \lvert A \rvert$	——

Tabla 4.6: Comparativa entre instrucciones aritméticas de la PROGRAMMA 101 y del MIPS R2000.

Otro punto muy diferente entre ambas son las instrucciones de transferencia de datos entre registros, donde la PROGRAMMA 101 dispone de cinco instrucciones frente a dos del MIPS R2000 —`mfhi` y `mflo`— y que se dedican a transferir datos de los dos registros dedicados a la multiplicación y división —cálculo entero—. Estas cinco instrucciones son las que principalmente le proporcionan a la PROGRAMMA 101 su versatilidad y potencial.

Para finalizar esta comparativa, se hace referencia a los saltos donde la PROGRAMMA 101 solo tiene una instrucción para el salto incondicional y una para el condicional, mientras que en el MIPS R2000 hay tres —`j`, `jal` y `jr`— y diez —`beq`, `bne`, `bgt`, `beqz`, ...— respectivamente, con distintos parámetros según las circunstancias.

CAPÍTULO 5

Diseño de programas

Ya que la Programma 101 disponía de un lenguaje ensamblador fácil de usar, Olivetti diseñó un formulario para que sirviera de plantilla a aquellos usuarios que se dedicaran a programar esta computadora. Dicha plantilla muestra de forma compacta toda la información necesaria para dicha función, tal como el contenido de los registros, tanto el código del programa como constantes necesarias, el estado de la rueda que configura el número de decimales, así como información para su archivo y asociación con otras tarjetas magnéticas donde se almacenara dicha programación. Esta plantilla se muestra en la figura 5.1 y en el apéndice C.

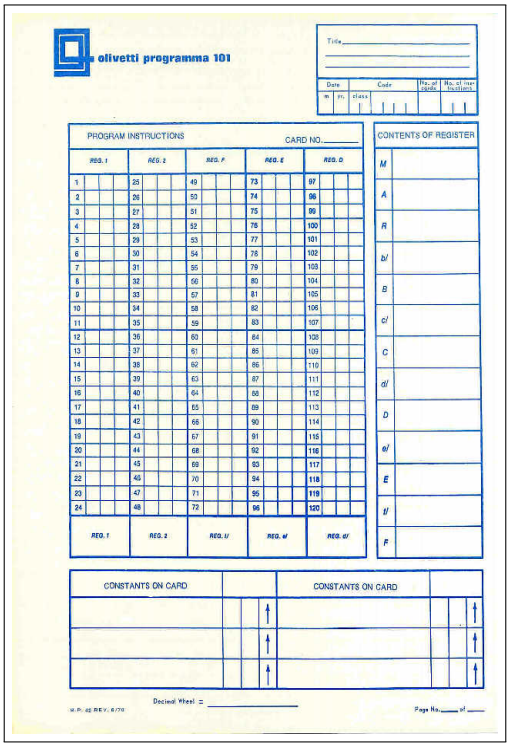


Figura 5.1: Plantilla [16] para programar la Programma 101.

A continuación se presenta la codificación de varios programas, así como una breve descripción de estos para poder constatar la simplicidad del lenguaje de la Programma 101 expuesto en el capítulo anterior. Todos y cada uno de los siguientes algoritmos se presentan en el apéndice D en sus correspondientes plantillas de programación, tal y como lo harían los programadores de la Programma 101 en los años en que esta estaba en todo su auge.

5.1 Factorial

Tras introducir por teclado el número que queremos factorizar (n), el programa nos devuelve el resultado de esta operación. Principalmente, el código presentado [10] consiste en un bucle que en cada iteración acumula la multiplicación del valor introducido y sus antecesores hasta que alcanza el valor cero, única condición existente en la Programma 101 para la evaluación en los saltos condicionales.

Este código —tabla 5.1— puede ser introducido en un único registro y presenta una situación de ejemplo en la que se introduce una constante en tiempo de ejecución. Además, se ha comentado cada línea para ayudar a esclarecer su funcionamiento.

1	**COMENTARIOS**
AV	→ *Inicio pulsando V*
S	→ *Introducción de n*
D↑	→ *Introducir n en el registro D*
↓	→ *Pasar n al acumulador*
AW	→ *Inicio del bucle de la multiplicación iterativa*
A/↑	→ *Comienzo de introducción de constante*
D/↓	→ *Constante* $= 1$, *fin de introducción*
-	→ $n = n - 1$
/V	→ *¿*$n > 0$*?*
D◇	→ *Si* $n = 0$, *imprime* $n!$
V	→ *Salto al comienzo*
A/V	→ *Si* $n > 0$ *entonces*
D↕	→ *Intercambia n con* $n!$ *(parcial)*
Dx	→ $n \times n!$ *(parcial)*
D↕	→ *Intercambia* $n!$ *(parcial) con n*
W	→ *Salto al inicio del bucle de la multiplicación iterativa*

Tabla 5.1: Código para calcular el factorial de un número en la Programma 101.

Para completar la comparativa del capítulo anterior sobre la memoria y las instrucciones del MIPS R2000, a continuación se expone el código en ensamblador necesario para realizar el cálculo del factorial de un número, en este caso el valor del entero de entrada está guardado en el registro `$4` y el resultado es almacenado en el `$2` —código comprobado mediante simulación con MARS ver. 4.4—.

```
        addi $2,$0,1        # $2 inicializa a 1
bucle:  blez $4,fin         # Si n-1 <= 0, entonces fin
        mul $2,$2,$4        # n! (parcial) * n-1 (parcial)
        addi $4,$4,-1       # n-1
        j bucle             # Salta a bucle
fin:    nop                 # Fin
```

A partir de las diferencias expuestas en el apartado 4.4, se puede observar que el código del MIPS R2000 tienen menos número de instrucciones para realizar el mismo cálculo, este no interactúa con el usuario.

Con la ayuda de un simulador —uno de los expuestos en el apartado 7.2— se procede a ejecutar el código de la tabla 5.1, obteniendo como resultados de calcular el factorial de 3, 4, 5 y 10 los que se muestran en la figura 5.2.

Figura 5.2: Simulación del código que calcula el factorial de un número.

5.2 Fibonacci

Dado un número n, este código nos devuelve el resultado de aplicar la función de Fibonacci para dicho valor. Para realizar esta operación es necesario guardar en dos registros unas constantes iniciales ($d = -1$ y $e = 1$) y realizar un bucle que calcula en cada iteración un elemento de la serie de Fibonacci y devuelve finalmente el resultado de la función. Las instrucciones que se muestran en la tabla siguiente están basadas en la programación que se muestra en el siguiente algoritmo iterativo escrito en pseudocódigo.

```
Programa
    a = -1
    b = 1
    n = leer de consola
    Para i de n hasta 0
        f = a + b
        a = b
        b = f
    Fin_Para
    Devuelve f
Fin_Programa
```

En este apartado también se ha incorporado una serie de comentarios al código para esclarecer el funcionamiento del algoritmo. Para facilitar la legibilidad se ha intentado recurrir a las instrucciones más relevantes dejando a un lado las relacionadas con las transferencias de datos entre registros y saltos incondicionales.

Este código —tabla 5.2— utiliza dos registros para almacenar el programa, aunque el segundo solo contiene tres instrucciones. Las constantes que se almacenan en los registros ”d” y ”e” pueden ser introducidas en tiempo de ejecución —de la misma forma que se han hecho en el apartado anterior— mediante las instrucciones A/↑, E/↓ (para -1) y A/↑, D/↓ (para 1) respectivamente.

1	**COMENTARIOS**
AV	
D/↓	
D↕	
E/↓	
E↕	
S	→ *Introducción de n*
C↑	→ *Introducir n en el registro C*
AY	→ *Inicio bucle de n hasta* 0
D↓	→
E+	→ } $f = a + b$
F↕	→
E↓	→
D↕	→ } $a = b$
F↓	→
E↕	→ } $b = f$
C↓	→ *Pasar n al acumulador*
/W	→ *¿n* > 0*?*
F◇	→
/◇	→ } *Si n = 0, imprime el resultado y una línea en blanco*
V	→ *Salto al comienzo.*
A/W	→ *Si n* > 0 *entonces*
A/↑	
D/↓	
-	→ $n = n - 1$
2	
C↕	
Y	
V	

Tabla 5.2: Código para calcular la función de Fibonacci de un número en la Programma 101.

Como en la exposición del apartado anterior, se realiza otra comparación entre el código ensamblador de la Programma 101 y el del MIPS R2000. A diferencia del caso anterior, el valor de entrada y el resultado son gestionados con la instrucción `syscall` que interactúa con el usuario —código comprobado mediante simulación con MARS ver. 4.4—.

```
addi $6,$0,-1        # a = $6 = -1
mul $5,$6,$6         # b = $5 = 1
li $2,5              # Código para lectura de entero
syscall              # Lee n
move $7,$4           # contador = $7 = n
```

```
bucle:   bltz $7, fin        # Si contador < 0, entonces fin
         add $2,$6,$5        # f = $2 = a + b
         move $6,$5          # a = b
         move $5,$2          # b = f
         addi $7,$7,-1       # contador -1
         j bucle             # Salta a bucle
fin:     li $2,1             # Código para imprimir entero
         syscall             # Lee n
         li $2,10            # Código para salir
         syscall             # Salir
```

En el código anterior se han omitido las instrucciones para gestionar el hilo de ejecución, ya que no se trata de un procedimiento dentro de otro código, sino un programa íntegro. Teniendo en cuenta que la PROGRAMMA 101 omite las instrucciones correspondientes a la introducción de las constantes en tiempo de ejecución, se ve una reducción en el número de instrucciones del MIPS respecto a la Olivetti. A medida que la complejidad del código aumenta, esta reducción se hace más evidente debido a las diferencias en el juego de instrucciones —amplitud y funcionalidad del MIPS R2000—.

Como en el apartado anterior y con ayuda del mismo simulador, se procede a ejecutar el código de la tabla 5.2 para calcular la función de Fibonacci de los valores 2, 5 y 6, obteniendo como resultados los que se muestran en la figura 5.3.

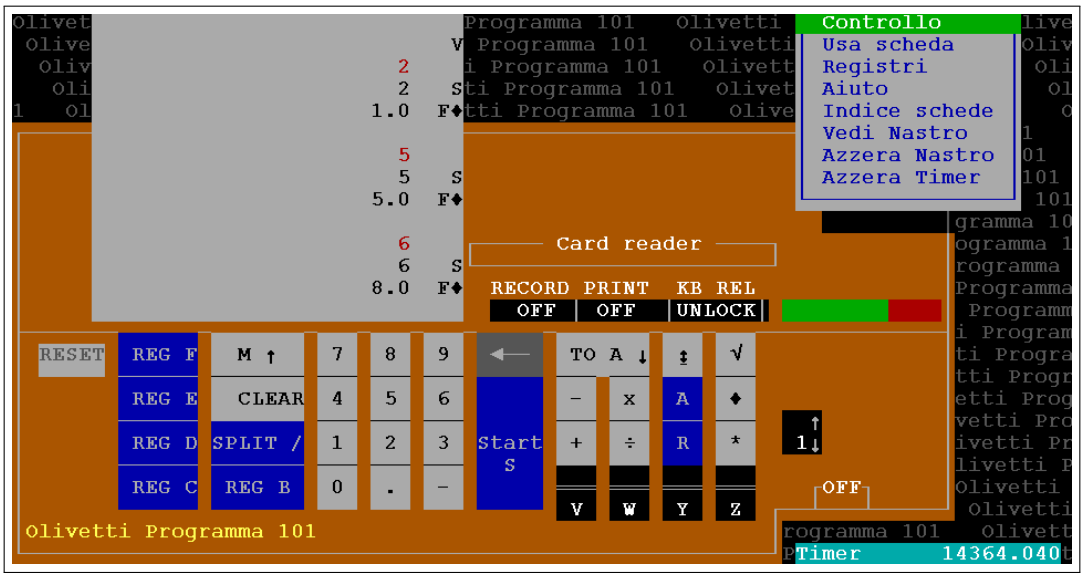


Figura 5.3: Simulación del código que calcula la función de Fibonacci de un número.

5.3 Ecuación de segundo grado

Para resolver la ecuación $ax^2 + bx + c = 0$ se introducen los valores correspondientes a los coeficientes a, b y c, y el código que se muestra en las tablas 5.3 y 5.4 [18] calcula la ecuación obteniendo el resultado de la variable x descompuesto en su parte real e imaginaria.

1	**COMENTARIOS**
AV	$\rightarrow$ *Inicio pulsando V*
S	$\rightarrow$ *Introducción de a*
B↑	
S	$\rightarrow$ *Introducción de b*
C↑	
S	$\rightarrow$ *Introducción de c*
↕	
BW	
B÷	$\rightarrow \dfrac{c}{a}$
B↕	
A+	$\rightarrow 2a$
C↕	
C÷	$\rightarrow$
A-	$\rightarrow$ $\left.\dfrac{-b}{2a}\right\}$
-	$\rightarrow$
A×	$\rightarrow (b/2a)^2$
C↑	
B-	$\rightarrow \beta$
/V	$\rightarrow$ *Si discriminante* > 0, *salta*
A-	
-	
/Z	$\rightarrow$ *Si discriminante* < 0, *salta*
C♢	$\rightarrow$ *Si discriminante* $= 0$, *entonces imprime* $x_1 = x_2$
V	$\rightarrow$ *Salto al comienzo*

Tabla 5.3: Código para resolver ecuaciones de segundo grado en la Programma 101 (registro 1).

2	COMENTARIOS
A/Z	$\rightarrow$ *Discriminante* < 0
A-	
-	
C$\diamondsuit$	$\rightarrow$ *Imprime la parte real de* x_1
A$\sqrt{\ }$	$\rightarrow$ *Calcula la parte imaginaria de* x_1
A$\diamondsuit$	$\rightarrow$ *Imprime la parte imaginaria de* x_1
C$\diamondsuit$	$\rightarrow$ *Imprime la parte real de* x_2
-	$\rightarrow$ *Calcula la parte imaginaria de* x_2
A$\diamondsuit$	$\rightarrow$ *Imprime la parte imaginaria de* x_2
V	$\rightarrow$ *Salto al comienzo*
A/V	$\rightarrow$ *Discriminante* > 0
A$\sqrt{\ }$	$\rightarrow \sqrt{\beta}$
C$\updownarrow$	
C+	$\rightarrow x_1$
A$\diamondsuit$	$\rightarrow$ *Imprime* x_1
C-	$\rightarrow$
-	$\rightarrow$ $\Big\}\, x_2$
A$\diamondsuit$	$\rightarrow$ *Imprime* x_2
V	$\rightarrow$ *Salto al comienzo*
AW	
C/$\downarrow$	
CW	

Tabla 5.4: Código para resolver ecuaciones de segundo grado en la PROGRAMMA 101 (registro 2).

Si el discriminante es mayor que 0, obtendremos dos resultados reales; si es igual a 0, la ecuación tiene una única raíz; y si es menor que 0, tiene dos soluciones, ambas con parte real e imaginaria. Los dos posibles valores del resultado que se han comentado se obtienen resolviendo las siguientes ecuaciones:

$$x_1 = -\frac{b}{2a} + \sqrt{\left(\frac{b}{2a}\right)^2 - \frac{c}{a}} \qquad x_2 = -\frac{b}{2a} - \sqrt{\left(\frac{b}{2a}\right)^2 - \frac{c}{a}}$$

$$\beta = \left(\frac{b}{2a}\right)^2 - \frac{c}{a} = \frac{b^2 - 4ac}{4a^2} \rightarrow b^2 - 4ac = Discriminante$$

Este código es un buen ejemplo para contemplar todas las operaciones aritméticas —menos el valor absoluto del acumulador— dentro del mismo programa, también se pueden observar varios saltos condicionales que sirven para decidir que resultado imprimir según el valor del discriminante.

En el cálculo que se ha expuesto, es necesario manejar números reales y raíces cuadradas, por ello, el MIPS R2000 por sí solo no puede calcular estas operaciones,

para conseguir este objetivo, se necesita la FPU MIPS R2010 —con ella se amplía el juego de instrucciones—, con su ayuda se consigue calcular la ecuación de segundo grado con el siguiente código ensamblador —código comprobado mediante simulación con MARS ver. 4.4—.

```
.data 0x10000000            # Para poder imprimir " "
espacio: .asciiz " "
.text

        li $2,6             # Código para lectura de real
        syscall             # Lee a
        mov.s $f1,$f0       # Guarda a en $f1
        syscall             # Lee b
        mov.s $f2,$f0       # Guarda b en $f2
        syscall             # Lee c
        li $2,2             # Código para impresión de real
        mov.s $f3,$f0       # Guarda c en $f3
        mul.s $f5,$f2,$f2   # $f5 = b^2
        li $5,4             # $5 = 4
        mtc1 $5,$f6         # Paso de enteros a reales
        cvt.s.w $f6,$f6     # $f6 = 4.0
        mul.s $f7,$f6,$f1   # $f6 = 4a
        mul.s $f7,$f7,$f3   # $f7 = 4ac
        sub.s $f5,$f5,$f7   # $f5 = b^2 - 4ac = Discriminante
        mul.s $f9,$f1,$f1   # $f9 =a^2
        mul.s $f6,$f6,$f9   # $f6 = 4a^2
        div.s $f6,$f5,$f6   # $f6 = Beta
        abs.s $f6,$f6       # $f6 = |Beta|
        add.s $f8,$f1,$f1   # $f8 = 2a
        div.s $f8,$f2,$f8   # $f8 = b/2a
        neg.s $f8,$f8       # $f8 = -b/2a
        c.eq.s $f5,$f20     # Si discriminante = 0
        bc1t igual          # Salta a igual
        c.lt.s $f5,$f20     # Si discriminante < 0
        bc1t menor          # Salta a menor
        bc1f mayor          # Si discriminante > 0, salta a mayor
igual:  mov.s $f12,$f8      # $f12 = x1 = x2
        syscall             # Imprime x1 = x2
        j fin               # Salta a fin
menor:  mov.s $f12,$f8      # $f12 = parte real de x1
        syscall             # Imprime la parte real de x1
        li $2,4             # Código para impresión de cadena
        la $4,espacio       # Cadena a imprimir
        syscall             # Imprime " "
        li $2,2             # Código para impresión de real
```

```
            sqrt.s $f6,$f6      # $f6 = parte imaginaria de x1
            mov.s $f12,$f6      # $f12 = parte imaginaria de x1
            syscall             # Imprime la parte imaginaria de x1
            li $2,4             # Código para impresión de cadena
            la $4,espacio       # Cadena a imprimir
            syscall             # Imprime " "
            li $2,2             # Código para impresión de real
            mov.s $f12,$f8      # $f12 = parte real de x2
            syscall             # Imprime la parte real de x2
            li $2,4             # Código para impresión de cadena
            la $4,espacio       # Cadena a imprimir
            syscall             # Imprime " "
            li $2,2             # Código para impresión de real
            neg.s $f12,$f6      # $f12 = parte imaginaria de x2
            syscall             # Imprime la parte imaginaria de x2
            j fin               # Salta a fin
mayor:      sqrt.s $f6,$f6      # $f6 = raiz cuadrada de Beta
            add.s $f12,$f8,$f6  # $12 = x1
            syscall             # Imprime x1
            li $2,4             # Código para impresión de cadena
            la $4,espacio       # Cadena a imprimir
            syscall             # Imprime " "
            li $2,2             # Código para impresión de real
            sub.s $f12,$f8,$f6  # $12 = x2
            syscall             # Imprime x2
            j fin               # Salta a fin
fin:        li $2,10            # Código para salir
            syscall             # Salir
```

El programa del MIPS R2000 expuesto anteriormente realiza las mismas funciones que el de la Programma 101 pero, como se puede observar, tiene más líneas de código, esto es debido a que la interacción con el usuario en la R2000 es más difícil de implementar. Además, para hacer la salida de datos más legible, se deben de realizar operaciones con direccionamiento en memoria —las tres primeras líneas hacen posible la impresión del caracter " "— y así imprimir caracteres, esto no se podía hacer con la Programma 101.

A continuación, se procede a simular el primer código para obtener una demostración gráfica —véase la figura 5.4— de su funcionamiento y de los resultados obtenidos al calcular la ecuación $x^2 + 2x + 5 = 0$.

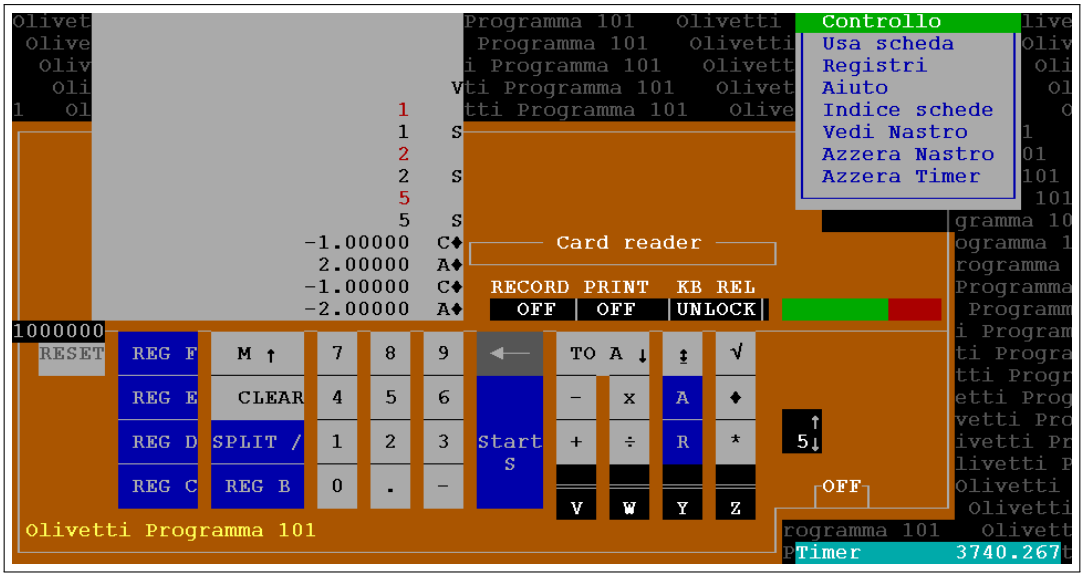


Figura 5.4: Simulación del código que calcula la ecuación de segundo grado.

5.4 Coseno

El siguiente código [18] —tabla 5.5— calcula la función $\cos(x)$, donde x se introduce por teclado como parámetro de entrada. En esta operación intervienen una serie de constantes que están almacenadas en el resto de registros; ”D”= 180, ”d”= −4,934745, ”E”= 4,058041, ”e”= −1,332369, ”F”= 0,22965 y ”f”= −0,020577.

Esta función se calcula utilizando la serie de Tchebycheff con cinco constantes o coeficientes, siendo su correspondencia con las almacenadas en los registros de la PROGRAMMA 101 la siguiente; $d = C_2$, $E = C_4$, $e = C_6$, $F = C_8$, $f = C_{10}$.

$$\cos(x) = 1 + C_2W^2 + C_4W^4 + C_6W^6 + C_8W^8 + C_{10}W^{10}$$

$$W = \frac{x}{D}$$

Entre las constantes está la almacenada en el registro ”D”, esta tiene un valor de 180, ya que este cálculo se realiza en grados sexagesimales, pero si se desea realizar en radianes solo hay que sustituir dicha constante por el valor de π; análogamente, si se desea el resultado en grados centesimales, el valor de este se debe sustituir por 200.

1	COMENTARIOS
AV	$\rightarrow$ *Inicio pulsando V*
S	$\rightarrow$ *Introducción de x*
$\downarrow$	
A$\updownarrow$	$\rightarrow \lvert x \rvert$
D$\div$	$\rightarrow W$
A$\times$	$\rightarrow W^2$
C$\updownarrow$	
F/$\downarrow$	$\rightarrow$
C$\times$	$\rightarrow$ $C_{10}W^2$
F+	$\rightarrow$
C$\times$	$\rightarrow$ $C_{10}W^4 + C_8W^2$
E/+	$\rightarrow$
C$\times$	$\rightarrow$ $C_{10}W^6 + C_8W^4 + C_6W^2$
E+	$\rightarrow$
C$\times$	$\rightarrow$ $C_{10}W^8 + C_8W^6 + C_6W^4 + C_4W^2$
D/+	$\rightarrow$
C$\times$	$\rightarrow$ $C_{10}W^{10} + C_8W^8 + C_6W^6 + C_4W^4 + C_2W^2$
A/$\uparrow$	$\rightarrow$ *Comienzo de introducción de constante*
D/$\downarrow$	$\rightarrow$ *Constante* $= 1$*, fin de introducción*
+	$\rightarrow \cos(x)$
A$\diamondsuit$	$\rightarrow$ *Imprime* $\cos(x)$
V	$\rightarrow$ *Salto al comienzo*

Tabla 5.5: Código para calcular el coseno de un número en la Programma 101.

A continuación y siguiendo con las comparativas, se muestra este algoritmo implementado en el ensamblador del MIPS R2000 —código comprobado mediante simulación con MARS ver. 4.4—, como se puede observar, más de la mitad de las instrucciones son para la introducción de datos e impresión del resultado —en la Programma 101 se usan las instrucciones "S" y "$\diamondsuit$"—, es posible una reducción de este tamaño si se utilizara otros simuladores como PCSPIM, en este se puede utilizar la pseudoinstrucción `li.s`, la cual introduce de forma implícita un número real en un determinado registro.

```
li $9,1000000
mtc1 $9,$f9              # Paso de enteros a reales
cvt.s.w $f9,$f9          # $f9 = 1000000
li $8,-4934745
mtc1 $8,$f2              # Paso de enteros a reales
cvt.s.w $f2,$f2          # $f2 = C2*1000000
div.s $f2,$f2,$f9        # $f2 = C2
li $8,4058041
mtc1 $8,$f4              # Paso de enteros a reales
```

```
cvt.s.w $f4,$f4          # $f4 = C4*1000000
div.s $f4,$f4,$f9        # $f4 = C4
li $8,-1332369
mtc1 $8,$f6              # Paso de enteros a reales
cvt.s.w $f6,$f6          # $f6 = C6*1000000
div.s $f6,$f6,$f9        # $f6 = C6
li $8,229650
mtc1 $8,$f8              # Paso de enteros a reales
cvt.s.w $f8,$f8          # $f8 = C8*1000000
div.s $f8,$f8,$f9        # $f8 = C8
li $8,-20577
mtc1 $8,$f10             # Paso de enteros a reales
cvt.s.w $f10,$f10        # $f10 = C10*1000000
div.s $f10,$f10,$f9      # $f10 = C10
li $2,6                  # Código para lectura de real
syscall                  # Lee x
mov.s $f1,$f0            # Guarda x en $f1
abs.s $f1,$f1            # $f1 = |x|
syscall                  # Lee D
mov.s $f3,$f0            # Guarda D en $f3
div.s $f3,$f1,$f3        # $f3 = W
mul.s $f3,$f3,$f3        # $f3 = W^2
mul.s $f5,$f10,$f3       # $f5 = C10*W^2
add.s $f5,$f5,$f8
mul.s $f5,$f5,$f3        # $f5 = C8*W^2+C10*W^4
add.s $f5,$f5,$f6
mul.s $f5,$f5,$f3        # $f5 = C6*W^2+C8*W^4+C10*W^6
add.s $f5,$f5,$f4
mul.s $f5,$f5,$f3        # $f5 = C4*W^2+C6*W^4+C8*W^6+C10*W^8
add.s $f5,$f5,$f2
mul.s $f5,$f5,$f3        # $f5 = C2*W^2+C4*W^4+C6*W^6+C8*W^8+C10*W^10
div.s $f9,$f9,$f9        # $f9 = 1
add.s $f5,$f5,$f9        # $f5 = cos(x)
mov.s $f12,$f5           # $f12 = $f5
li $2,2                  # Codigo para imprimir real
syscall                  # Imprime cos(x)
li $2,10                 # Codigo para salir
syscall                  # Salir
```

El código expuesto incorpora una variación respecto al de la PROGRAMMA 101, en este caso, además de introducir el ángulo, el usuario debe proporcionar el valor de ”D” y así especificar si el resultado se quiere en grados sexagesimales, radianes o en grados centesimales.

Con uno de los simuladores descritos en el apartado 7.2 —de la misma forma que se ha hecho en los apartados anteriores—, se ha procedido a simular este algoritmo, obteniéndose el resultado que se puede ver en la figura 5.5 para los valores de 0, 90, 180 y −56.

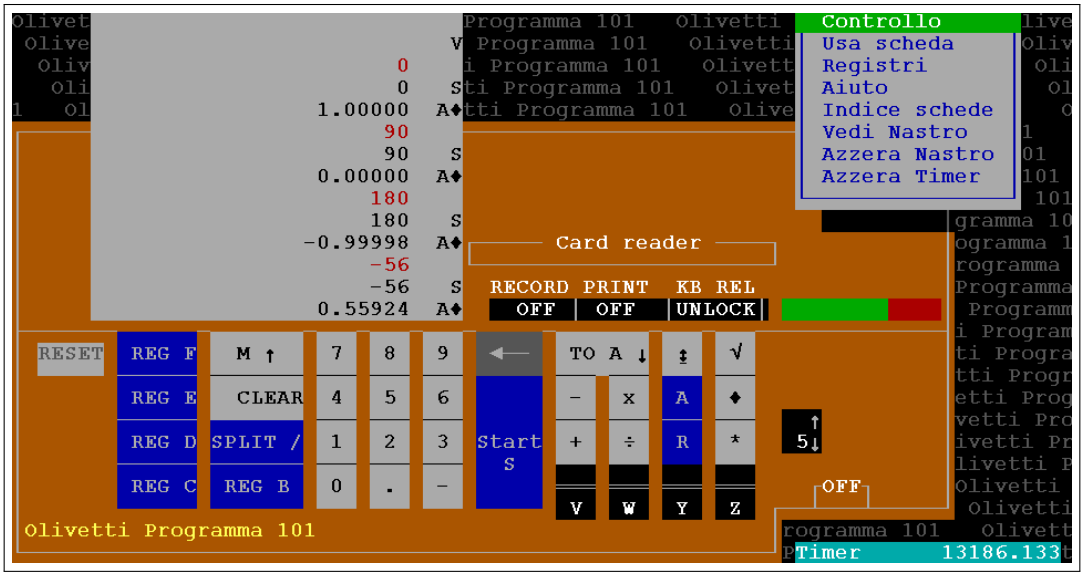


Figura 5.5: Simulación del código que calcula el coseno de un ángulo.

5.5 Aterrizaje lunar

La Programma 101 surgió en plena carrera espacial entre Estados Unidos —misiones Vanguard y Apolo— y la Unión Soviética —misiones Sputnik y Vostok—, este contexto social favorecía, y la empresa Olivetti fomentaba, que esta máquina se relacionara con el campo de la astronomía.

El código que se muestra en la tabla 5.6 [3] devuelve los valores de velocidad, altura y combustible disponible partiendo de un estado inicial desde el cual se le va introduciendo de forma paulatina las cantidades de combustible consumida, el programa recalcula y responde nuevamente con los tres parámetros mencionados.

El comportamiento de este programa corresponde en un aumento en la velocidad de descenso con cantidades pequeñas de consumo de combustible y, por lo contrario, en una reducción de dicha velocidad a medida que se aumenta el consumo; además nos indica la distancia que queda para alcanzar la superficie lunar.

1	2	d	D	e	E	f	F
AV	↓				A/↑		C↓
A/↑	A↕				R*		C/↕
D-	D↕				R*		B↓
E/↑	D↓				R*		B/↕
A/↑	D/-				R*		D↓
RS	/V				R*		D/↕
E-	A↕				E*		CV
B/↑	D↕				♢		A/V
A/↑	E/-				S		A/↑
RS	C↕						D↑
RS	C↓						↓
D-	B/+						E/×
C/↑	B↕						C/×
A/↑	C↓						D↕
RS	A/↑						B/↓
R↑	D↑						B/×
D↓	÷						D+
D/↑	B/+						A√
BV	C/+						B/↕
B/♢	C↕						BW
C/♢	C↓						B/♢
D/♢	/W						A/↑
/♢	CW						DS
S	A/W						♢

Tabla 5.6: Código para simular un aterrizaje lunar en la Programma 101.

Este algoritmo devuelve cero en sus tres parámetros de salida cuando el aterrizaje ha sido perfecto, en caso contrario se considera que la operación ha finalizado con una colisión proporcionando un dato negativo de velocidad, altura igual a cero y en combustible aparece el valor $-999\,999$; si el valor de la velocidad es muy próxima a cero se puede concluir que la colisión ha sido leve y no supone peligro para los tripulantes de la nave simulada.

En la figura 5.6 se puede visualizar un ejemplo de ejecución realizado en el mismo simulador que se ha utilizado en este capítulo, en ella se muestra el comienzo y el final del programa.

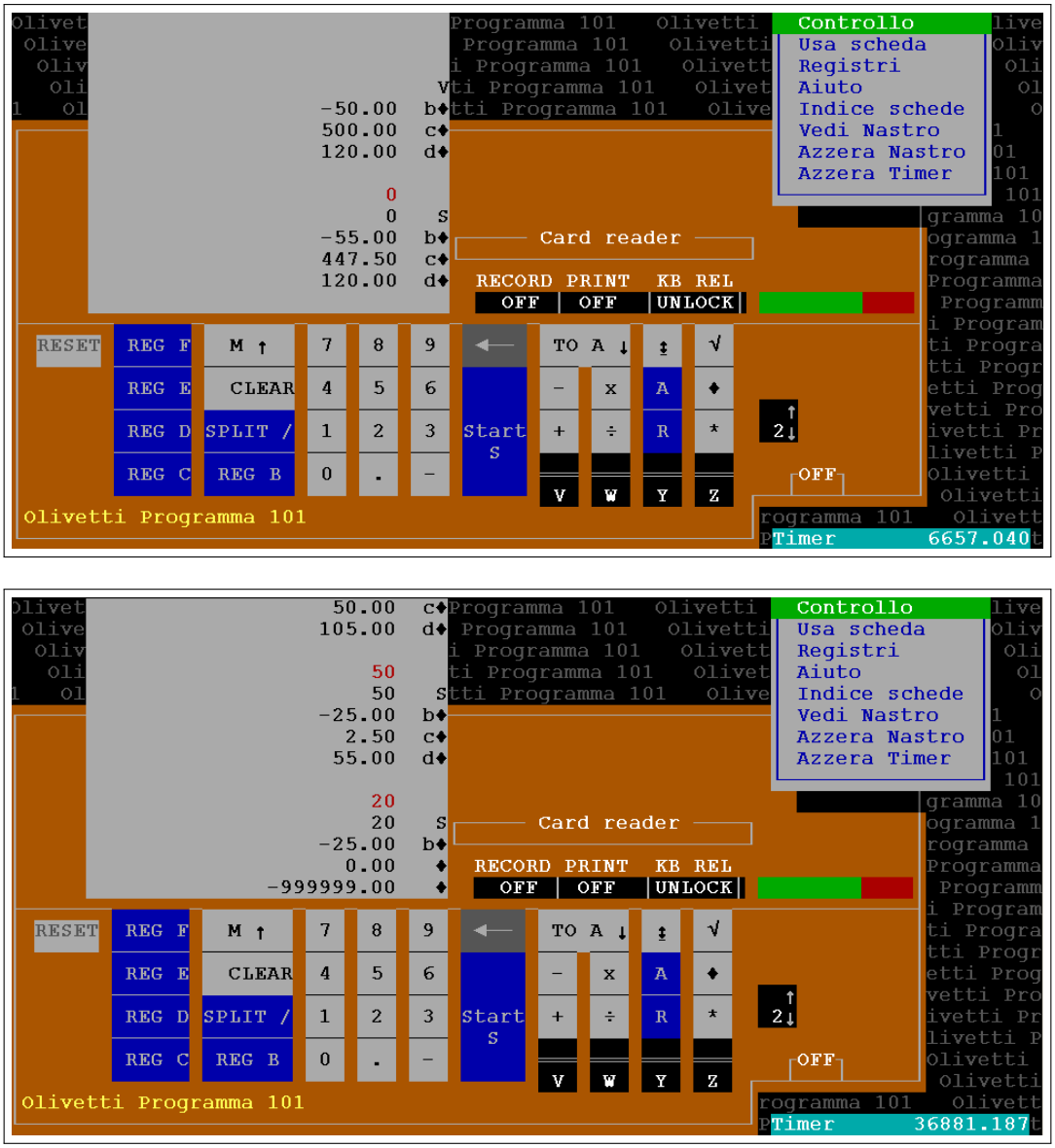


Figura 5.6: Inicio y final de la simulación de aterrizaje lunar.

5.6 Órbita de un satélite

El código de la tabla 5.7 [3] fue el utilizado en la presentación oficial de la Programma 101 en la Exposición Universal de Nueva York del año 1965 y entusiasmó al público que allí estaba presente. Consiste en el cálculo de la órbita de un satélite sobre un plano (x, y) basándose en las siguientes ecuaciones [1][1]:

$$m \times \frac{dv_x}{dt} = \frac{-GM \times m \times x}{r^3} \qquad m \times \frac{dv_y}{dt} = \frac{-GM \times m \times y}{r^3} \qquad r = \sqrt{x^2 + y^2}$$

[1]Marca de tiempo: 11 segundos.

Se considera el producto $GM = 1$, es decir, las distancias están dadas en megametros —un millón de metros—. Para iniciar el programa hay que introducir un punto (x e y) y las velocidades (v_x y v_y) iniciales, así como un valor de epsilón (dt, en segundos) y una cantidad que indica el número de ciclos que debe realizar el programa para que aparezca una impresión con la posición del satélite (x e y) que va recorriendo el satélite.

1	2	d	D	e	E	f	F
AV	Ax						S
S	×						S
B↑	E/↕						AZ
S	E↓						B/+
B/↑	B×						B/↕
S	E/÷						D↓
C↑	D↕						C↕
S	C↓						D/↓
C/↑	D-						C/↕
S	D↕						F/↓
E↑	E↓						Y
S	B/×						S
F↑	E/÷						S
BV	D/↕						S
F↓	C/↓						AY
A/V	D/-						A/↑
F/↕	D/↕						D/↓
B↓	E↓						-
B×	D×						/V
E/↕	B+						B♢
B/↓	B↕						B/♢
B/×	E↓						/♢
E/+	D/×						CV
A√	Z						S

Tabla 5.7: Código para calcular la órbita de un satélite alrededor de la Tierra en la Programma 101.

Al realizar una prueba de ejecución, tal y como se hizo en la presentación oficial de la Programma 101, con valores iniciales de uno para epsilón y para los ciclos por impresión, así como unos parámetros de entrada[2] tales como los que se muestran a continuación:

$$x = 0 \qquad v_x = 0{,}4$$

$$y = 8 \qquad v_y = -0{,}0078$$

[2] `http://www.astronomia.net/cosmologia/lec122.htm`

Como ocurrió en su presentación, la PROGRAMMA 101 devuelve una lista de resultados —valores de x e y—, en aquella época los asistentes a la demostración debían analizar aquellos resultados numéricos, pero actualmente dichos resultados pueden ser trasladados a un software para representarlos en una gráfica obteniendo, tras 350 datos —coordenadas— de salida, la órbita que se muestra en la figura 5.7.

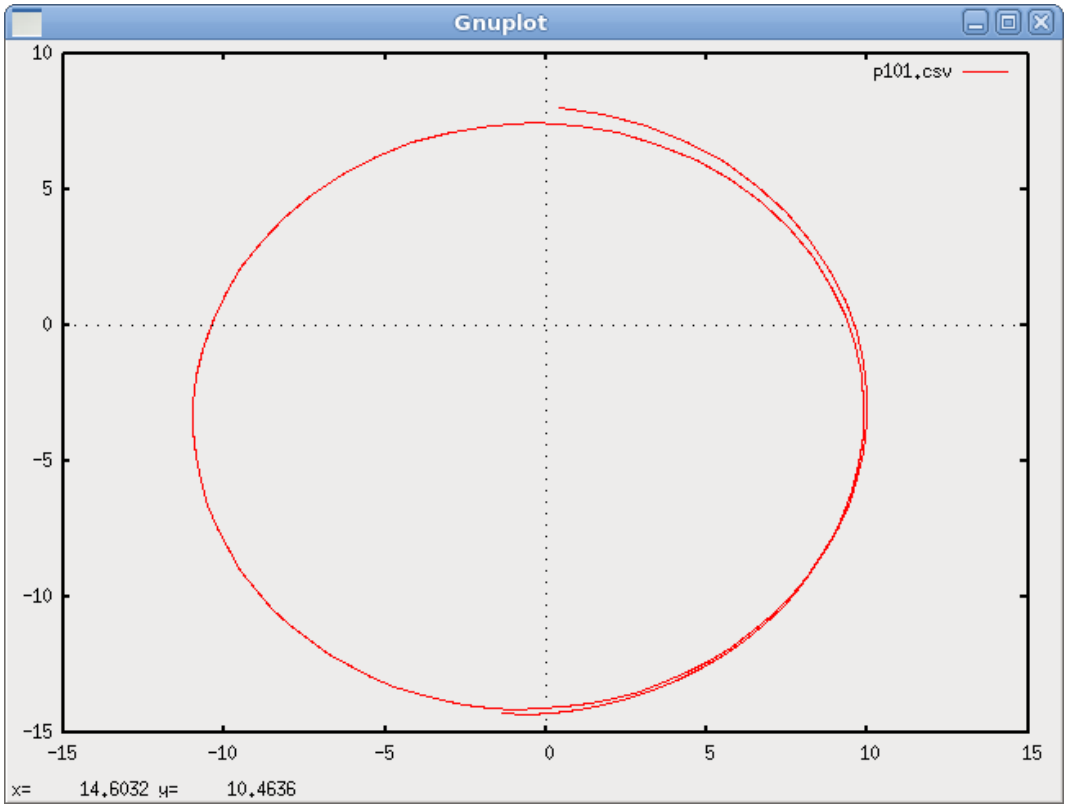


Figura 5.7: Gráfica del resultado obtenido en el cálculo de la órbita de un satélite con la PROGRAMMA 101.

Para obtener los resultados anteriores, se ha realizado la simulación que se muestra en la figura 5.8, y como en todas las simulaciones anteriores, el simulador guarda un registro con los datos de entrada y salida llamado `P101.PRN`, a partir de él se crea un fichero `CSV`, y de este, se genera la gráfica anterior.

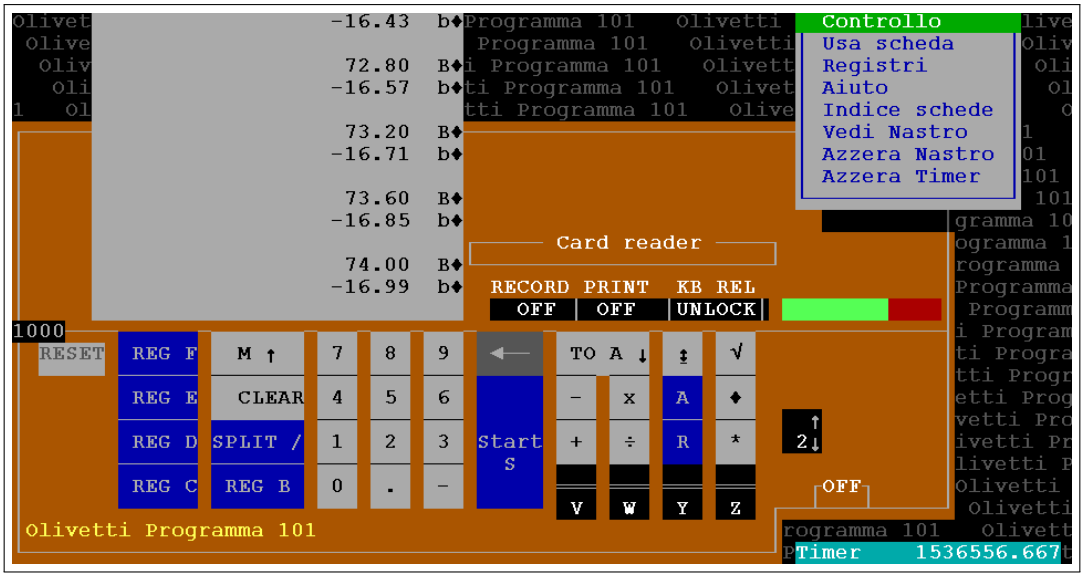


Figura 5.8: Simulación del código que calcula la órbita de un satélite alrededor de la Tierra.

CAPÍTULO 6

La calculadora HP 9100

En 1968, cuatro años después de la aparición de la Programma 101, surgió la calculadora programable HP 9100, producida y comercializada por Hewlett Packard y cuyo diseño se inspiró en el del producto de Olivetti, desbancando a esta del mercado. En este capítulo se procede a describir la HP 9100 y se ponen en relación ambas calculadoras.

La HP 9100 —véase en la figura 6.1—, igual que la Programma 101, fue construida con electrónica discreta, por lo que pertenece a la segunda generación —también por el año en el que apareció—. Además, tiene una carcasa de aluminio fundido con unas dimensiones de $40{,}64 \times 48{,}26 \times 20{,}95$ cm y un peso de 18 kilogramos, es decir, más pequeña y ligera que la Programma 101.

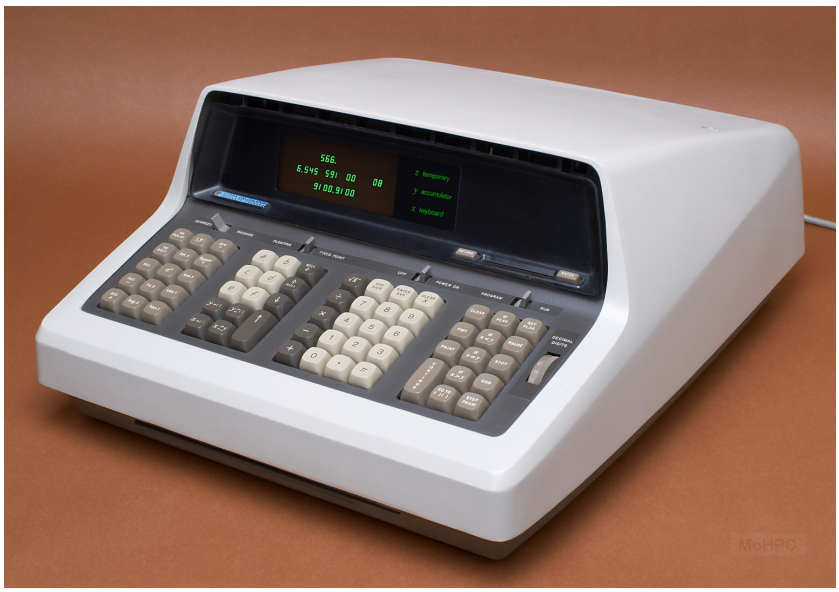


Figura 6.1: Calculadora HP 9100 [30].

6.1 Memoria: tecnología y distribución

La memoria en la HP 9100 esta formada por núcleos magnéticos —toros— como los que se muestran en la figura 6.2. Los toros de ferrita se disponen en una matriz bidimensional de modo que son atravesadas por dos hilos, X e Y, que discurren según las filas y columnas.

Para escribir un bit en la memoria se envía un pulso simultáneamente por las líneas X_i e Y_j correspondientes al toro situado en la posición (i, j), el cual se magnetizará en el sentido dado por los pulsos. Los demás toros, tanto de la fila como de la columna, no varían su magnetización ya que solo reciben un pulso —X o Y—, siendo su campo magnético insuficiente para vencer la histéresis del toro.

El dato se lee mediante un nuevo hilo Z, que recorre todos los toros de la matriz. Escribimos un cero por el método descrito anteriormente, luego únicamente el toro (i, j) puede cambiar de estado. Si contiene un cero, no cambia, luego en la línea Z no se tiene señal; pero si el toro tiene un uno, pasa a valer cero, su sentido de magnetización cambia e induce un pulso en la línea Z, que se leerá como uno. Como se ve, el proceso descrito destruye el dato que se lee, luego en las memorias formadas por toros es necesario reescribir el dato tras leerlo.

A medida que la memoria aumentaba en capacidad hubo que pasar a una disposición tridimensional, apilando capas de toros y utilizando las mismas líneas X e Y, por ello se añadió un hilo de inhibición para seleccionar el plano de ferritas a las que se deseaba acceder.

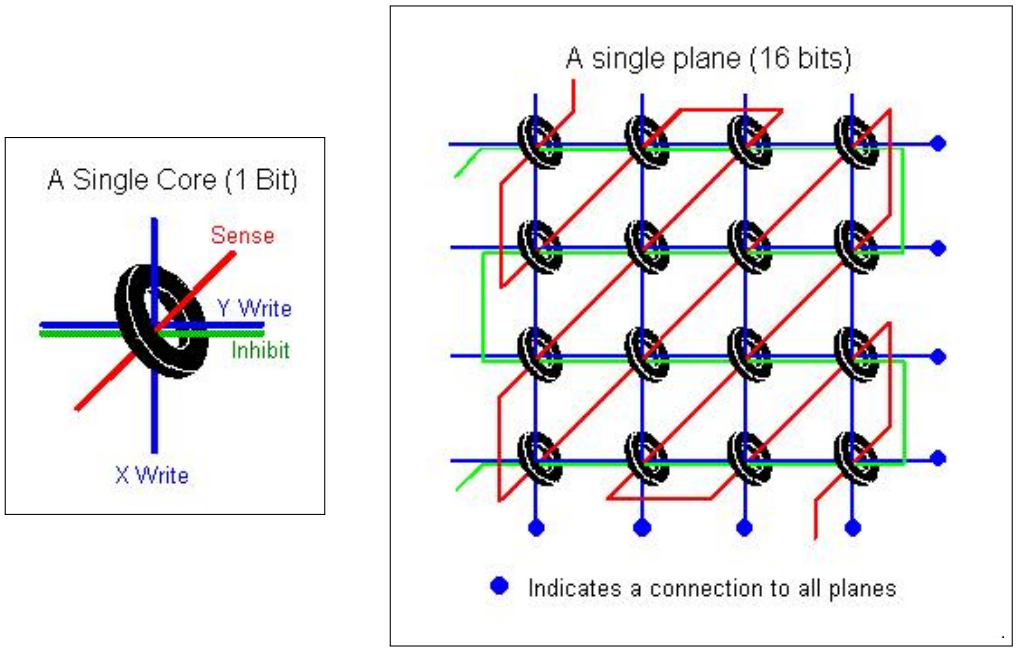


Figura 6.2: Núcleo magnético y su disposición en un plano [30].

Una de las propiedades de los núcleos magnéticos descritos es que mantienen la información almacenada aún en ausencia de alimentación eléctrica, por lo que el usuario podría apagar la calculadora sin perder los datos o programas almacenados, algo que la Programma 101 no podía hacer.

Utilizando esta tecnología que se ha explicado, la HP 9100 disponía de memoria ROM (*Read-Only Memory*) y RAM, la primera tenía una capacidad de 32 Kib [1] —512 palabras de 64 bits— y contenía los programas o rutinas necesarios para el arranque inicial de la calculadora, así como funciones matemáticas programadas de fábrica. La RAM tenía 2 208 bits de memoria, más del doble que la Programma 101 (960 bits), organizados como $6 \times 16 \times 23$ bits.

Respecto a la distribución de la memoria, hay una gran similitud con la que nos encontramos en la Programma 101. La HP 9100 almacenaba la información en formato BCD —4 bits por cada dígito— y tiene un total de 19 registros divididos de la siguiente forma:

- **3 registros:** Los registros X, Y y Z son los análogos a los registros de la Programma 101 M, A y R respectivamente.
- **2 registros:** Los registros E y F se utilizan para almacenar exclusivamente datos, igual que los registros B y C de la Programma 101.
- **14 registros:** Enumerados del 0 al 9, A, B, C y D pueden guardar tanto datos como instrucciones, como ocurría en la Programma 101 con los registros D, E y F. Cada registro dispone de 14 dígitos para datos o instrucciones dando un total de 196 instrucciones, superior a los 120 de la Programma 101.

Como se puede observar en este apartado, la distribución de los registros y sus funcionalidades tienen una gran similitud con las encontradas en la Programma 101, aunque esta última disponía de 2 registros exclusivos para instrucciones que la HP 9100 no dispone.

6.2 Manejo y programación

El modo en que se utiliza la HP 9100 es diferente a la Programma 101 en varios aspectos; por un lado, en su teclado hay dos teclas más grandes que el resto, una es "↑" —"ENTER (up)"— para la introducción de datos y la otra "CONTINUE" para la validación o confirmación —como "ENTER" en los ordenadores actuales—; por otra parte, la notación que utiliza es la RPN (*Reverse Polish Notation*), esta representa expresiones matemáticas sin necesidad de utilizar paréntesis siendo equivalentes "$(9-5) \div 2$" y "9 5 − 2 ÷".

Esta calculadora puede funcionar en modo manual o programada, según la opción activada en el selector "RUN/PROGRAM". Para programarla, primero hay que inicializarla colocando el selector mencionado en "RUN" y pulsando "END" —pone

[1] ISO - International Organization for Standardization. ISO/IEC 80000-13:2008

el contador de programa a 00—, a continuación se pone el selector en ”PROGRAM”; luego, se van introduciendo las instrucciones mientras en el *display* se visualizan en el registro X como un código octal. La tabla 6.1 muestra la relación entre esos códigos y sus correspondientes teclas.

COD.	TECLA		COD.	TECLA
00	0		40	$y \rightarrow ()$
01	1		41	Stop
02	2		42	Fmt
03	3		43	If Flag
04	4		44	Go To () ()
05	5		45	Print/Space
06	6		46	End
07	7		47	Continue
10	8		50	If $x = y$
11	9		51	
12	e		52	If $x < y$
13	a		53	If $x > y$
14	b		54	Set Flag
15	f		55	$\lvert y \rvert$
16	c		56	π
17	d		57	Pause
20	Clear		60	Acc+
21	.		61	Rcl
22	Roll up		62	To Polar
23	$x \rightarrow ()$		63	Acc−
24	$y \leftrightarrows ()$		64	Int x
25	drop		65	$\ln x$
26	Enter Exp		66	To Rect
27	Enter (up)		67	Hyper or $x \leftarrow ()$
30	$x \leftrightarrows y$		70	$\sin x$
31	Roll dn		71	$\tan x$
32	Chg Sign		72	Arc
33	+		73	$\cos x$
34	−		74	e^x
35	÷		75	$\log x$
36	×		76	$\sqrt{}$
37	Clear x		77	Sub or Return

Tabla 6.1: Relación entre los códigos y las instrucciones de la HP 9100.

Una vez introducido el programa, se vuelve a poner el selector en ”RUN” y se inicializa el contador de programa —a un valor 00— pulsando la tecla ”END”, finalmente se procede a su ejecución pulsando la tecla ”CONTINUE”.

6.3 Interfaces y periféricos

La HP 9100 tiene un teclado y una rueda para configurar el número de posiciones decimales —como la Programma 101— formando la interfaz de entrada y una pantalla CRT (*Cathode Ray Tube*) donde cada dígito tenía un formato de 8 segmentos como interfaz de salida. Esta última sustituye en funcionalidad a la impresora de la Programma 101, y en caso de que el usuario quisiera imprimir información de salida, la HP 9100 disponía de una impresora opcional que se compraba aparte y se acoplaba en la parte superior de su carcasa.

Un periférico que incorporaba la HP 9100 de serie era el lector/grabador de tarjetas magnéticas, un dispositivo que solo se había visto antes en la Programma 101; en este caso las tarjetas ideadas por HP tenían otra forma menos alargada, tal y como se muestra en la figura 6.3.

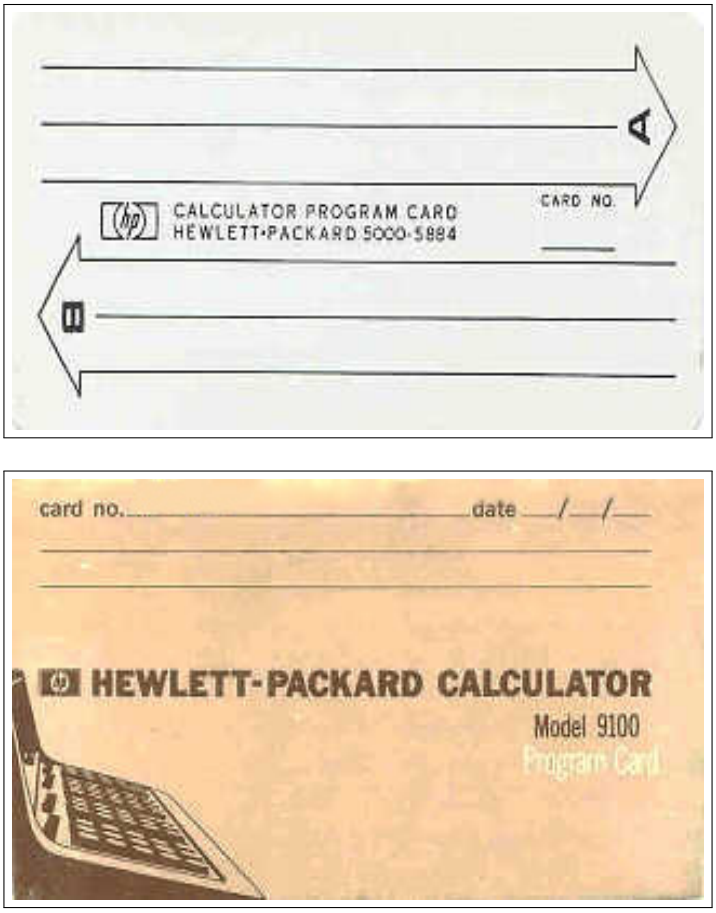


Figura 6.3: Tarjeta magnética de la HP 9100[2].

[2] `http://www.oldcalculatormuseum.com`

Hewlett Packard creó alrededor de la HP 9100 una gama de periféricos que se compraban por separado como es el caso de la impresora, que ya se ha mencionado, plóteres, digitalizadores, lectores ópticos de tarjetas, pantallas grandes, así como ampliaciones de memoria y otros. A partir de la primera versión, la HP 9100A, aparecieron nuevas versiones donde se incorporaban mejoras como mayor capacidad de memoria, inclusión de un puerto de comunicaciones serie y otras.

Tras la descripción que se ha realizado en este capítulo sobre la calculadora HP 9100, se hace evidente que el diseño de esta calculadora comparte muchos puntos con la Programma 101, y por ello Olivetti recibió de Hewlett Packard el pago de regalías.

CAPÍTULO 7

Recursos en Internet

Durante el desarrollo de este trabajo se ha podido comprobar la existencia de gran cantidad de recursos en Internet sobre la Programma 101, desde sitios web oficiales destinados a la conservación y difusión de su historia, hasta webs y blogs mantenidos por aficionados a la retroinformática, todo ello, pasando por sitios web de propósito general y webs que han sido desarrollados por los propios protagonistas de la creación de la máquina objeto de estudio de este documento.

7.1 Wikipedia y webs de retroinformática

Como referencia de partida y para consultar los aspectos más genéricos de la Programma 101 siempre se puede acceder, por un lado, a las páginas web de la Wikipedia, donde podemos encontrar información en varios idiomas como español, inglés e italiano —figura 7.1—, entre otros. Por otro lado, a los sitios y blogs de los aficionados a la retroinformática mostrándose unos ejemplos en las figuras 7.2 y 7.3.

Figura 7.1: Página web de la Wikipedia que trata sobre la Programma 101.

(a) `http://www.curtamania.com`

(b) `http://www.silab.it`

Figura 7.2: Páginas webs de aficionados a la retroinformática que tratan sobre la Programma 101.

(a) `http://old8bits.blogspot.com.es`

(b) `http://royal.pingdom.com`

Figura 7.3: Blogs de aficionados a la retroinformática que tratan sobre la Programma 101.

La afición a la retroinformática, y más concretamente a la Olivetti Programma 101, llega en algunos casos a crear movimientos como el llamado «Proyecto 101» —«*101 Project*»— cuya página web se puede ver en la figura 7.4. Esta iniciativa está dedicada a la Programma 101, recopilando recuerdos de personas que han estado relacionadas con ella, opiniones, vídeos y otros materiales en distintos formatos, y se comparten experiencias o vivencias con ella y su entorno. Por ejemplo, una persona que fue comercial de Olivetti expresa en esta plataforma el hecho de haber recibido en 1967 un curso de programación de la Programma 101 para desempeñar su trabajo.

Figura 7.4: `http://www.101project.eu`

Entre todos los recursos encontrados en Internet sobre la Programma 101, se ha de hacer especial mención a dos webs en particular. Una de ellas, la creada en honor a Pier Giorgio Perotto, mantenida por su hijo Pierpaolo Perotto. En el transcurso de la creación de este trabajo ha estado inaccesible durante un tiempo para luego volver a estar activa con un aspecto renovado, mostrándose finalmente tal y como se puede ver en la figura 7.5.

Figura 7.5: `http://www.piergiorgioperotto.it`

La segunda web de mención especial es la correspondiente a la *Associazione Archivio Storico Olivetti*, en la cual se expone información en diversos formatos sobre la historia de la empresa Olivetti en todos sus ámbitos como puede ser los personajes, los productos, la publicidad, entre otros. En la figura 7.6 se puede observar la página principal de esta web.

Figura 7.6: `http://www.storiaolivetti.it`

7.2 Simuladores

De entre todos los recursos que se han localizado en Internet, hay un tipo que es de gran utilidad para hacerse una idea de cómo era manejar la Programma 101: se trata de los simuladores. En Internet se pueden encontrar varios simuladores sobre esta máquina, pero en este trabajo se han seleccionado dos de ellos debido a sus características.

El primero de estos simuladores es el desarrollado por Marco Galeotti [10]. Tiene una interfaz didáctica enfocada a enseñar cómo funcionaba internamente la Programma 101, no mantiene el aspecto de esta pero es muy útil para comprender cómo funciona el flujo de datos dentro de la máquina. Este simulador consta de tres secciones claramente diferenciadas tanto visualmente como en funcionamiento se refiere:

- **Editor de ficheros:** En esta sección podemos leer y guardar programas de/a ficheros externos con extensión `.101` y otros tipos de ficheros de texto plano. Además, este panel dispone de un conjunto de botones que nos permite realizar operaciones de edición sobre los ficheros mencionados.
- **Editor de programa:** Este editor nos permite crear programas instrucción a instrucción fácilmente, ya que los botones de este tienen un color y están

dispuestos según su funcionalidad. Con este panel se puede añadir comentarios a cada línea de código del programa editado.

- **Simulador de ejecución:** Ejecuta las instrucciones del programa introducido. Esta ejecución se inicia pulsando las teclas V, W, Y o Z según requiera el programa, y el modo de ejecución puede ser continuo o paso a paso. Este último es muy útil para conocer con detalle el funcionamiento de la Programma 101, puesto que a través de unos campos nos muestra en todo momento el contenido de sus registros.

A continuación se muestran las figuras 7.7 y 7.8, donde podemos ver la web de Marco Galeotti y su simulador que ha sido objeto de artículos en revistas especializadas en retroinformática [11] y ha sido amablemente cedido por su desarrollador a través de correo electrónico.

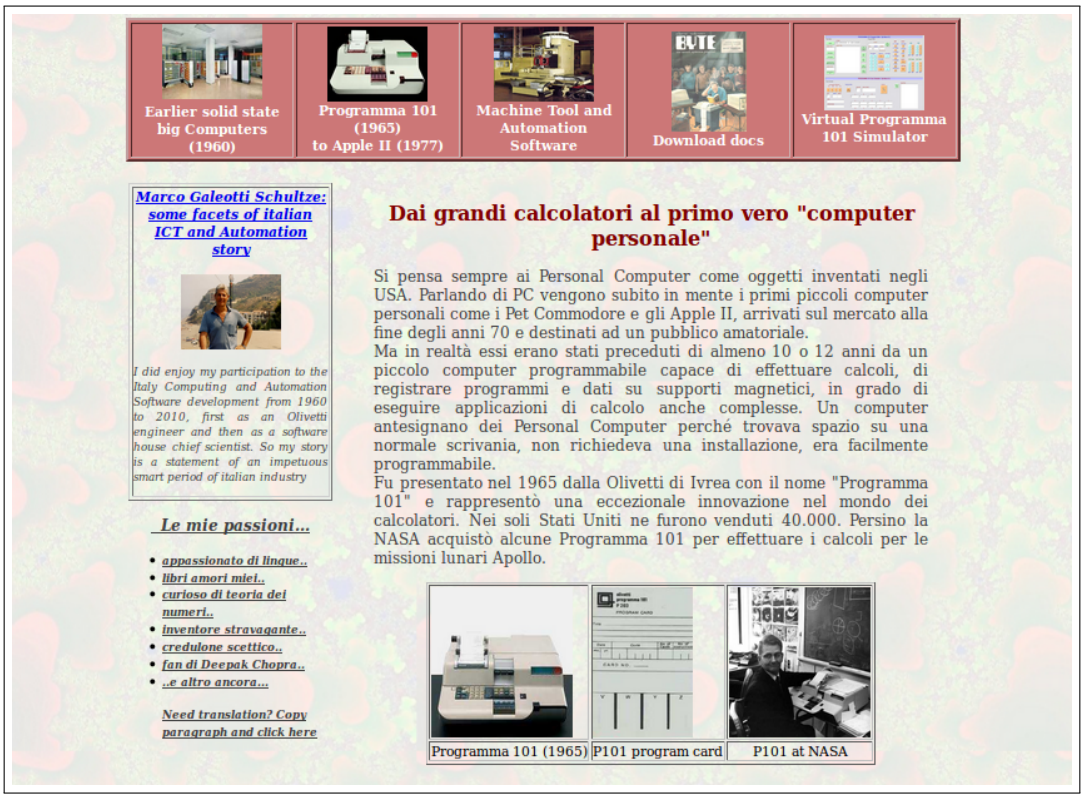


Figura 7.7: Web de Marco Galeotti
`http://www.marcogaleotti.com/Calcolatori.html`.

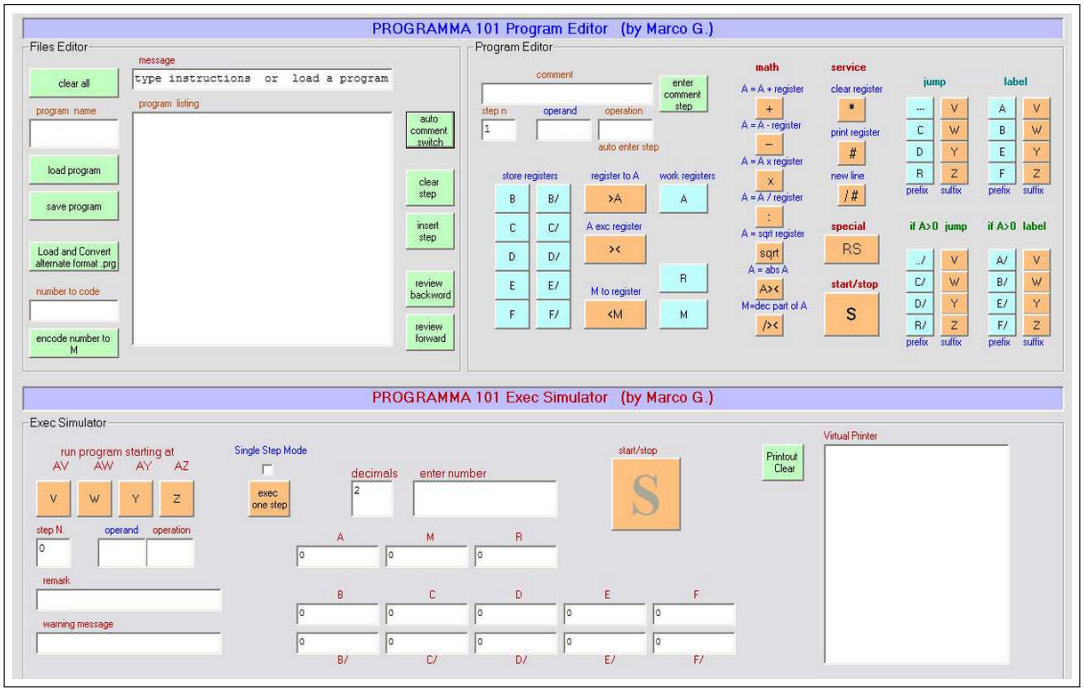


Figura 7.8: Simulador desarrollado por Marco Galeotti.

El segundo de los simuladores seleccionados es el desarrollado por el ingeniero Claudio Larini [3]. Este simulador, a pesar de tener una interfaz espartana, ha sido seleccionado porque reproduce el teclado, salida de datos y otros controles tal y como nos lo encontraríamos delante de una Programma 101, tanto en disposición ante el usuario como en proporciones entre ellos.

El interfaz que presenta este simulador es muy similar al de la Programma 101 pero, además, el modo en que se usa también lo es, si seguimos los manuales de usuario [20] y de programación [19] lo podemos manejar sin problemas.

Además, este simulador incorpora en su esquina inferior derecha un temporizador que presenta una medida del tiempo de ejecución realista de la Programma 101 expresada en milisegundos, y en su esquina superior derecha un pequeño menú donde se puede cargar un programa en concreto, ver el estado de los registros, una ayuda, un índice de los programas que guarda y podemos cargar, ver las anotaciones de los eventos que van sucediendo y, por último, borrar o reiniciar dichas anotaciones y el temporizador.

Esta aplicación fue desarrollada inicialmente en QuickBasic 4.5 para MS-DOS, y posteriormente en QB64 para adaptarla a Windows 7 y 8. Además incorpora un ejecutable muy interesante llamado `P101TOQB.EXE` que permite convertir un código escrito en el lenguaje de la Programma 101 a un código en BASIC —archivos `.BAS`—.

El simulador de Claudio Larini también ha sido objeto de artículos en revistas especializadas en retroinformática [9] y se puede ver a continuación —en las figuras 7.9 y 7.10— junto a la web de su desarrollador.

EMULATORE PER OLIVETTI PROGRAMMA 101

OLIVETTI PROGRAMMA 101 EMULATOR

Dopo aver letto una recensione del libro dell'Ing. **Perotto** "*Programma 101. L'invenzione del personal computer: una storia appassionante mai raccontata*", ho iniziato ad interessarmi della storia della P101 della quale, nonostante ne sia quasi contemporaneo (nel 1965 avevo cinque anni), non avevo mai sentito parlare. Il risultato finale, oltre a raccogliere oltre 600 mega di manuali, fotografie, video, brevetti e documentazione varia, è stato realizzare un emulatore completo, per poter eseguire e provare il software.

L'emulatore è a 16 bit e funziona con il mouse: potrebbe essere caricato in modalità autoavviante su un vecchio portatile (486 o Pentium) ed eventualmente una stampantina termica con interfaccia parallela da almeno 28 colonne così da farlo funzionare come una P101.

L'emulatore è stato scritto in Quick Basic 4.5 con una libreria di funzioni BCD, il che permette di ottenere gli stessi risultati della macchina originale. Dopo aver scoperto l'esistenza di **QB64**, progetto *open-source* che consente di portare in ambienti 32/64 bit programmi scritti per Qbasic/QuickBasic senza doverli riscrivere completamente, l'emulatore è stato modificato per poter essere compilato sotto QB64 ed è nato così il "fratellino", **P101EMUL64.EXE**, che può essere eseguito nativamente anche sotto Windows 7 & 8.

Oltre al programma principale (**P101EMUL.EXE**) è disponibile pure **P101TOQB.EXE** che permette di tradurre un programma P101 in QuickBasic, generando il file .BAS corrispondente. Tale programma accetta fino a 400 istruzioni P101, permettendo così di eseguire e verificare anche programmi che non potrebbero girare sulla P101 originale (limitata, com'è noto, a 120 istruzioni).

Nella sezione ***DOCUMENTAZIONE*** più in basso è possibile scaricare vari manuali della

Figura 7.9: Web de Claudio Larini
`http://www.claudiolarini.altervista.org/emul2.htm`.

Figura 7.10: Simulador desarrollado por Claudio Larini.

CAPÍTULO 8

Material didáctico

El Museo de Informática de la Escuela Técnica Superior de Ingeniería Informática de la Universitat Politècnica de València se inauguró oficialmente el 11 de diciembre de 2001. El 13 de mayo de 2013 fue reconocido oficialmente por la Conselleria d'Educació, Cultura i Esport como museo oficial de la Comunidad Valenciana, y finalmente el Consejo Ejecutivo del ICOM-España [5] ha aprobado que el Museo de Informática de la UPV forme parte de esta organización internacional. En la figura 8.1 se muestra la placa acreditativa como museo oficial de la Comunidad Valenciana y el logotipo del ICOM.

Figura 8.1: Placa acreditativa como museo oficial de la Comunidad Valenciana y logotipo del organismo internacional ICOM.

La finalidad de este museo es difundir y promover en la sociedad el conocimiento del patrimonio digital que se ha generado a través de los tiempos, y hacer recapacitar sobre su función en la sociedad a lo largo de su historia, presente y del que puede tener en el futuro.

Este trabajo final de grado pretende ayudar al Museo de Informática a que cumpla con su misión mediante la incorporación de una página web dentro de la sección «Artículos: El museo te cuenta» del sitio web de dicho museo [13]. Para ello se incluyen textos explicativos, imágenes, vídeos y enlaces a material relacionados con la Programma 101, como por ejemplo simuladores.

La inclusión de esta información en la web se ha realizado en colaboración con el personal del museo encargado a tal efecto respetando los formatos y estilos ya existentes en dicha web, y quedando una página como la que se muestra en la figura 8.2.

Figura 8.2: Página web creada en castellano para el Museo de Informática sobre la Programma 101.

El Museo de Informática pertenece a los museos oficiales de la Comunidad Valenciana, por lo tanto, se ha procedido a crear una segunda página web idéntica a la mencionada pero en este caso traducida al valenciano, lengua cooficial de esta comunidad, dando como resultado una página como la que aparece en la figura 8.3.

DESENVOLUPAMENT I PRESENTACIÓ

En la primavera de 1962 Roberto Olivetti, el llavors president de la companyia italiana Olivetti, va donar instruccions a l'enginyer Pier Giorgio Perotto per a iniciar l'estudi de viabilitat d'una calculadora electrònica capaç d'automatitzar una seqüència d'instruccions que hauria d'estar a l'abast d'un usuari no especialitzat, a un preu raonable i que tinguera unes dimensions reduïdes paregudes a les d'una màquina d'escriure.

Pier Giorgio Perotto va idear una màquina, xicoteta, flexible, amb memòria, fàcil d'usar i programable per mitjà d'un llenguatge de programació senzill perquè qualsevol persona poguera programar-la, bàsicament era com un **ordinador personal.**

Per a exercir l'encàrrec, Perotto va crear un equip de desenvolupament amb quatre hòmens més; Gastone Garziera, Giovanni De Sandre, Giancarlo Toppi i Giuliano Gaiti.

Prema l'imatge per a ampliar

L'equip de desenvolupament amb el segon prototip de la Programma 101. (Agost de 1964)

D'esquerra a dreta i de dalt a baix: Gastone Garziera, Giancarlo Toppi, Pier Giorgio Perotto i Giovanni De Sandre. Giuliano Gaiti no apareix perquè va ser qui va realitzar la fotografia amb la càmera de Perotto.

Cortesia d'Associazione Va veure Storico Olivetti, Ivrea, Itàlia.

Dos punts molt importants per a aconseguir l'èxit del projecte va ser, d'una banda, substituir les memòries existents en el moment per **memòries de línia de retard acústica magnetostrictiva**, ja que era una bona opció per les seues dimensions, senzillesa i cost. D'altra banda, es varen utilitzar els nous (en aquell moment) **transistors 2N708** que junt amb la seua mida reduïda no tenien tants problemes de sobrecalfament i fiabilitat com les vàlvules de buit que s'utilitzaven fins al moment, a més que la seua vida útil era excel·lent.

El treball de l'equip de desenvolupament de la Programma 101 va finalitzar la seua tasca a l'abril de 1964 –dos anys de treball– aconseguint un producte que seria el començament de la història dels PC (Personal Computer) o ordinadors personals i que van batejar, per motius de

Figura 8.3: Página web creada en valenciano para el Museo de Informática sobre la Programma 101.

8.1 Estructura de las páginas web

Las dos páginas web desarrolladas a partir de este documento tienen el mismo contenido pero en idiomas diferentes —castellano y valenciano— y su organización sigue la estructura que se explica en este apartado.

Tras una presentación de la Programma 101, se ha procedido a relatar de donde surgió la idea de crearla, quién formaba el equipo de desarrollo, qué retos tenían que superar estos y cómo fue presentada ante la sociedad en la Exposición Universal de Nueva York.

Como la aparición de «el primer ordenador personal de sobremesa» obtuvo gran expectación, se presenta una muestra de la prensa norteamericana y de la publicidad que Olivetti lanzó para aprovechar aquel momento. Además, se mencionan dos aplicaciones en el que la Programma 101 intervino, la Guerra de Vietnam y la misión de la NASA Apolo 11.

A partir de este punto, las páginas web toman un aspecto más técnico, en ellas se describe la tecnología empleada en la construcción de esta máquina y especialmente en la memoria, se explica cómo eran las novedosas tarjetas magnéticas y su importancia en los medios de almacenamiento actuales y cómo se involucró Mario Bellini en su creación diseñando una carcasa agradable —estéticamente hablando— y ergonómica.

A continuación, se explica cómo está organizada la memoria y registros, se muestra una tabla con el juego de instrucciones y se presenta una plantilla de programación cumplimentada tal y como lo hacían los programadores de la Programma 101 en la época de su aparición.

Finalmente, se exponen dos simuladores, uno más didáctico desarrollado por Marco Galeotti y el otro, con una interfaz más realista, por Claudio Larini, ambos presentan una captura de pantalla y sus correspondientes enlaces a las webs de sus desarrolladores.

En las dos páginas web se han incluido imágenes y vídeos para aumentar la comprensión de lo expuesto en ellas, así como enlaces a sitios web donde se puede ampliar la información de los apartados correspondientes.

CAPÍTULO 9

Conclusiones

Este trabajo ha consistido en un estudio dedicado a la Olivetti Programma 101, abarcando sus aspectos histórico y técnico, dando una visión de quién, dónde y cómo fue creada, los retos y logros alcanzados en ese proceso, así como detalles de su arquitectura interna y funcionamiento. En él se ha pretendido alcanzar los objetivos que se describieron en el primer capítulo de la siguiente forma:

1. Se ha enmarcado históricamente la creación de la Programma 101, por quién, en qué circunstancias y la repercusión que alcanzó su aparición en aquella época.

2. Se han descrito los componentes con los que fue construida, su organización, y se ha explicado el juego de instrucciones que disponía para su programación.

3. Se han mostrado varios ejemplos de algoritmos codificados con sus instrucciones para demostrar el grado de dificultad que esta máquina tenía para los usuarios no familiarizados con la informática.

4. Tanto desde el punto de vista de la arquitectura interna como desde la perspectiva del juego de instrucciones, se ha comparado la Programma 101 con el procesador MIPS R2000, y de esa forma se ha podido tomar una referencia en dichos ámbitos.

5. En el desarrollo de este documento han tomado un papel primordial los recursos en Internet utilizados, entre estos se han destacado dos simuladores que facilitan la comprensión y describen su funcionamiento.

6. De todo el material recopilado en este trabajo se han creado las correspondientes páginas en el sitio web del Museo de Informática para difundir y preservar la información recogida en este documento.

He de mencionar que a la Programma 101 se le llama también *Perottina* en honor a Pier Giorgio Perotto. Sin embargo, en este trabajo no se ha utilizado este término para dejar patente que la creación de esta máquina fue un trabajo en equipo y que no todo el mérito fue de Perotto. De hecho, hay dos miembros del equipo —Toppi y Gaiti— que históricamente han sido olvidados en muchas fuentes de información, pero aquí se han adjuntado referencias donde sí se les asocia a dicho proyecto.

Para la realización de este documento he tenido que afrontar dos aspectos necesarios para poder concluirlo; uno es el aprendizaje del lenguaje LaTeX para poder crear el trabajo escrito; otro es la comunicación con personas de Italia vía correo electrónico mediante el idioma inglés, ya que yo no conozco el italiano y mis interlocutores no saben castellano.

9.1 Líneas de futuro

A partir de este trabajo fin de grado se pueden plantear diversos futuros proyectos, que como este, pueden ayudar al Museo de Informática en su labor de difusión y preservación del patrimonio digital. Ejemplos de estos pueden ser el estudio de otros computadores y elementos electrónicos mencionados en este trabajo como el IBM 360, el primer microprocesador Intel 4004 o el ordenador de navegación del Apolo 11 (AGC, *Apollo Guidance Computer*) —misión en la que participó la Programma 101—. Por último, a partir del simulador del ingeniero Claudio Larini se puede proponer el estudio, modificación y/o actualización del conversor `P101TOQB.EXE` para conseguir el paso de código de la Programma 101 a otro lenguaje distinto al QuickBasic, e incluso que pudiera realizar la conversión inversa.

En el año 2011 había ocho Programma 101 en funcionamiento, y este documento ha pretendido subsanar el desconocimiento existente del impacto que esta tuvo en el desarrollo de los sistemas informáticos personales y darle la importancia que se merece, ya que sentó las bases de todos los aspectos referentes a la informática personal y consiguió aportar una interesante alternativa a los entornos tan restrictivos que habían en su época.

Bibliografía

[1] Archivio Storico Istituto Luce. *Italia - Presentato il calcolatore elettronico da tavolo.* Radar R0037, 1966. Disponible en <`http://www.archivioluce.com/archivio`> [Consultado el 10 de febrero de 2015].

[2] Carlos A. Coello Coello. *Breve historia de la computación y sus pioneros.* México: Fondo de Cultura Económica, 2003.

[3] Claudio Larini. *Emulatore per Olivetti Programma 101.* Disponible en <`http://web.tiscali.it/claudiolarini/emul2.htm`> [Consultado el 24 de octubre de 2014].

[4] David A. Patterson y John L. Hennessy. *Estructura y diseño de computadores: la interfaz software/hardware.* Barcelona: Reverté, 2011.

[5] ICOM España | Consejo Internacional de Museos. Disponible en <`http://www.icom-ce.org`> [Consultado el 2 de junio de 2015].

[6] Isaías Pérez Pérez y Citlali Anahí Monzalvo López. *Introducción a la Arqueología Informática.* Hidalgo, México: Instituto de Ciencias Básicas e Ingeniería. Universidad Autónoma del Estado de Hidalgo, 2011.

[7] John L. Hennessy y David A. Patterson. *Organización y diseño de computadores: la interfaz hardware/software.* Madrid: Mcgraw-Hill, 1994.

[8] John von Neumann. *First Draft of a Report on the EDVAC.* Disponible en <`https://web.archive.org/web/20130314123032/http://qss.stanford.edu/\~godfrey/vonNeumann/vnedvac.pdf`> [Consultado el 17 de junio de 2015].

[9] Lorenzo2. "Emulare l'Olivetti P101." *Jurassic news - Retrocomputer Magazine*, Año 7, número 39, pp. 28-33, enero 2012.

[10] Marco Galeotti. *Dai grandi calcolatori al primo vero "computer personale".* Disponible en <`http://www.marcogaleotti.com/Calcolatori.html`> [Consultado el 26 de septiembre de 2014].

[11] Marco Galeotti. "La Programma 101, un oggetto troppo in anticipo sui tempi." *Jurassic news - Retrocomputer Magazine*, Año 9, número 51, pp. 5-12, noviembre 2014.

[12] Michael R. Williams. *A history of computing technology.* Los Alamitos, CA: IEEE Society Press, segunda edición, 1997.

[13] Museo de Informática | Escuela Técnica Superior de Ingeniería Informática | Universidad Politécnica de Valencia. Disponible en <`http://museo.inf.upv.es`> [Consultado el 20 de enero de 2015].

[14] Nadia Penserini. *Elementi di programmazione per calcolatore Olivetti programma 101.* Bologna, Italia: Nicola Zanichelli Editore, 1973.

[15] Olivetti Programma 101. Disponible en <`http://www.silab.it/frox/p101/_index.html`> [Consultado el 21 de octubre de 2014].

[16] Olivetti Programma 101 "Perottina". Disponible en <`http://www.curtamania.com/curta/database/brand/olivetti/Olivetti%20Programma%20101/index.html`> [Consultado el 26 de septiembre de 2014].

[17] Olivetti Programma 101 - Wikipedia. Disponible en <`https://it.wikipedia.org/wiki/Olivetti_Programma_101`> [Consultado el 24 de septiembre de 2014].

[18] Olivetti. *Programma 101: Calcolatore elettronico da tavolo, biblioteca programmi.* Ivrea, Italia: Olivetti Corp., volume 1, 1965.

[19] Olivetti. *Programma 101: Calcolatore elettronico da tavolo, manuale di programmazione.*. Ivrea, Italia: Olivetti Corp., 1965.

[20] Olivetti. *Programma 101: Calcolatore elettronico da tavolo, manuale generale.* Ivrea, Italia: Olivetti Corp., 1965.

[21] Olivetti. *Programma 101: General reference manual.* Ontario, Canada: Olivetti Canada Limited, 1965.

[22] Olivetti, Storia di un'impresa. Associazione Archivio Storico Olivetti, Ivrea, Italia. Disponible en <`http://www.storiaolivetti.it`> [Consultado el 6 de febrero de 2015].

[23] Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura. Disponible en <`http://www.unesco.org/new/es`> [Consultado el 21 de octubre de 2014].

[24] Paul E. Ceruzzi. *A history of modern computing.* Londres, Inglaterra: The MIT Press, segunda edición, 2003.

[25] Philippe Breton. *Historia y crítica de la informática.* Madrid: Cátedra, 1989.

[26] Pier Giorgio Perotto. Disponible en <http://www.piergiorgioperotto.it> [Consultado el 1 de junio de 2015].

[27] Pier Giorgio Perotto. *Programma 101 - L'invenzione del personal computer: Una storia appassionante mai raccontata.* Cuneo, Italia: Sperling & Kupfer Editori, 1995.

[28] Pier Giorgio Perotto, Giovanni De Sandre. "Program controlled electronic computer." U.S. Patent 3 495 222, 10 de febrero, 1970. Disponible en <http://www.google.com/patents/US3495222> [Consultado el 21 de enero de 2015].

[29] Programma 101 - Museo Tecnologicamente. Disponible en <http://www.museotecnologicamente.it/collezione/programma-101> [Consultado el 21 de noviembre de 2014].

[30] The Museum of HP Calculators. Disponible en <http://www.hpmuseum.org/hp9100.htm> [Consultado el 7 de enero de 2015].

[31] Zenit Arti Audiovisive - Docabout 2011. *Programma 101: La máquina que cambió el mundo.* History channel, 2012. Disponible en <https://www.youtube.com/watch?feature=player_detailpage&v=HkI04PkTZTc> [Consultado el 23 de enero de 2015].

APÉNDICE A

Carta para la preservación del Patrimonio Digital

La UNESCO en su función de conservación, progreso y difusión del saber, velando por la conservación y la protección del patrimonio universal, redactó una carta enfocada al patrimonio digital debido a su relevancia para la humanidad.

La carta que se muestra en este apéndice consta de un preámbulo y doce artículos repartidos en cuatro secciones, teniendo la siguiente estructura:

- PREÁMBULO.
- EL PATRIMONIO DIGITAL COMO HERENCIA COMÚN.
  Artículos 1 y 2.
- VIGILANCIA CONTRA LA PÉRDIDA DE PATRIMONIO.
  Artículos del 3 al 5.
- MEDIDAS NECESARIAS.
  Artículos del 6 al 9.
- ATRIBUCIONES.
  Artículos del 10 al 12.

CARTA PARA LA PRESERVACIÓN DEL PATRIMONIO DIGITAL

PREÁMBULO

La Conferencia General,

Considerando que la desaparición de cualquier forma de patrimonio empobrece el acervo de todas las naciones,

Recordando que la Constitución de la UNESCO establece que la Organización "[debe ayudar] a la conservación, al progreso y a la difusión del saber, velando por la conservación y la protección del patrimonio universal de libros, obras de arte y monumentos de interés histórico o científico", que su Programa Información para Todos ofrece una plataforma para el debate y la acción sobre políticas de información y sobre la salvaguardia de los conocimientos conservados en forma documental, y que su programa "Memoria del Mundo" tiene por objeto garantizar la preservación del patrimonio documental del mundo y un acceso universal al mismo,

Reconociendo que esos recursos de información y expresión creativa se elaboran, distribuyen, utilizan y conservan cada vez más en forma electrónica, y que ello da lugar a un nuevo tipo de legado: el patrimonio digital,

Consciente de que el acceso a dicho patrimonio brindará mayores oportunidades de creación, comunicación e intercambio de conocimientos entre todos los pueblos,

Entendiendo que este patrimonio digital se encuentra en peligro de desaparición, y que su preservación en beneficio de las generaciones actuales y futuras es una preocupación urgente en el mundo entero,

Proclama los siguientes principios y aprueba la presente Carta.

EL PATRIMONIO DIGITAL COMO HERENCIA COMÚN

Artículo 1 - Alcance

El patrimonio digital consiste en recursos únicos que son fruto del saber o la expresión de los seres humanos. Comprende recursos de carácter cultural, educativo, científico o administrativo e información técnica, jurídica, médica y de otras clases, que se generan directamente en formato digital o se convierten a éste a partir de material analógico ya existente. Los productos "de origen digital" no existen en otro formato que el electrónico.

Los objetos digitales pueden ser textos, bases de datos, imágenes fijas o en movimiento, grabaciones sonoras, material gráfico, programas informáticos o páginas Web, entre otros muchos formatos posibles dentro de un vasto repertorio de diversidad creciente. A menudo son efímeros, y su conservación requiere un trabajo específico en este sentido en los procesos de producción, mantenimiento y gestión.

Muchos de esos recursos revisten valor e importancia duraderos, y constituyen por ello un patrimonio digno de protección y conservación en beneficio de las generaciones actuales y futuras.

Este legado en constante aumento puede existir en cualquier lengua, cualquier lugar del mundo y cualquier campo de la expresión o el saber humanos.

Artículo 2 - Acceso al patrimonio digital

El objetivo de la conservación del patrimonio digital es que éste sea accesible para el público. Por consiguiente, el acceso a los elementos del patrimonio digital, especialmente los de dominio público, no debería estar sujeto a requisitos poco razonables. Al mismo tiempo, debería garantizarse la protección de la información delicada o de carácter privado contra cualquier forma de intrusión.

Los Estados Miembros tal vez deseen trabajar en colaboración con las organizaciones e instituciones pertinentes para propiciar un contexto jurídico y práctico que maximice la accesibilidad del patrimonio digital. Convendría reafirmar y promover un justo equilibrio entre los derechos legítimos de los creadores y otros derechohabientes y el interés del público por tener acceso a los elementos del patrimonio digital, de conformidad con las normas y los acuerdos internacionales.

VIGILANCIA CONTRA LA PÉRDIDA DE PATRIMONIO

Artículo 3 - El peligro de pérdida

El patrimonio digital del mundo corre el peligro de perderse para la posteridad. Contribuyen a ello, entre otros factores, la rápida obsolescencia de los equipos y programas informáticos que le dan vida, las incertidumbres existentes en torno a los recursos, la responsabilidad y los métodos para su mantenimiento y conservación y la falta de legislación que ampare estos procesos.

Los cambios en las conductas han ido a la zaga del progreso tecnológico. La evolución de la tecnología digital ha sido tan rápida y onerosa que los gobiernos e instituciones no han podido elaborar estrategias de conservación oportunas y bien fundamentadas. No se ha comprendido en toda su magnitud la amenaza que pesa sobre el potencial económico, social, intelectual y cultural que encierra el patrimonio, sobre el cual se edifica el porvenir.

Artículo 4 - Necesidad de pasar a la acción

A menos que se haga frente a los peligros actuales, el patrimonio digital desaparecerá rápida e ineluctablemente. El hecho de estimular la adopción de medidas jurídicas, económicas y técnicas para salvaguardar ese patrimonio redundará en beneficio de los propios Estados Miembros. Urge emprender actividades de divulgación y promoción, alertar a los responsables de formular políticas y sensibilizar al gran público tanto sobre el potencial de los productos digitales como sobre los problemas prácticos que plantea su preservación.

Artículo 5 - Continuidad del patrimonio digital

La continuidad del patrimonio digital es fundamental. Para preservarlo se requerirán diversas medidas que incidan en todo el ciclo vital de la información digital, desde su creación hasta su utilización. La preservación a largo plazo del patrimonio digital

empieza por la concepción de sistemas y procedimientos fiables que generen objetos digitales auténticos y estables.

MEDIDAS NECESARIAS

Artículo 6 - Elaborar estrategias y políticas

Es preciso elaborar estrategias y políticas encaminadas a preservar el patrimonio digital, que tengan en cuenta el grado de urgencia, las circunstancias locales, los medios disponibles y las previsiones de futuro. La colaboración de los titulares de derechos de autor y derechos conexos y otras partes interesadas a la hora de definir formatos y compatibilidades comunes, así como el aprovechamiento compartido de recursos, pueden facilitar esa labor.

Artículo 7 - Seleccionar los elementos que deben conservarse

Al igual que ocurre con el conjunto del patrimonio documental, los principios de selección pueden diferir de un país a otro, aun cuando los principales criterios para determinar los elementos digitales dignos de conservación sean su significado y valor duraderos en términos culturales, científicos, testimoniales o de otra índole. Indudablemente, se deberá dar prioridad a los productos "de origen digital". Los procesos de selección y de eventual revisión subsiguiente han de llevarse a cabo con toda transparencia y basarse en principios, políticas, procedimientos y normas bien definidos.

Artículo 8 - Proteger el patrimonio digital

Los Estados Miembros han de disponer de mecanismos jurídicos e institucionales adecuados para garantizar la protección de su patrimonio digital.

Hacer que la legislación sobre archivos, así como el depósito legal o voluntario en bibliotecas, archivos, museos u otras instituciones públicas de conservación, se aplique al patrimonio digital, ha de ser un elemento esencial de la política nacional de preservación.

Convendría velar por el acceso a los elementos del patrimonio digital legalmente depositados, dentro de límites razonables, sin que ese se haga en perjuicio de la explotación normal de esos elementos.

Para prevenir la manipulación o modificación deliberada del patrimonio digital, es de suma importancia disponer de un marco tanto jurídico como técnico en el que se proteja la autenticidad. Esto exige, en ambos casos, mantener los contenidos, el funcionamiento de los ficheros y la documentación en la medida necesaria para garantizar que se conserva un objeto digital auténtico.

Artículo 9 - Preservar el patrimonio cultural

Por definición, el patrimonio digital no está sujeto a límites temporales, geográficos, culturales o de formato. Aunque sea específico de una cultura, cualquier persona del mundo es un usuario en potencia. Las minorías pueden dirigirse a las mayorías y los individuos a un público de dimensión mundial.

Hay que preservar y poner a disposición de cualquier persona el patrimonio digital de todas las regiones, naciones y comunidades a fin de propiciar, con el tiempo, una representación de todos los pueblos, naciones, culturas e idiomas.

ATRIBUCIONES

Artículo 10 - Funciones y atribuciones

Los Estados Miembros tal vez deseen designar a uno o más organismos que se encarguen de coordinar la preservación del patrimonio digital y poner a su disposición los recursos necesarios. La división de tareas y atribuciones puede basarse en las funciones y competencias existentes.

Convendría adoptar medidas para:

a) instar a los fabricantes de equipos y programas informáticos, creadores, editores y productores y distribuidores de objetos digitales, así como otros interlocutores del sector privado, a colaborar con bibliotecas nacionales, archivos y museos, y otras instituciones que se ocupen del patrimonio público, en la labor de preservación del patrimonio digital;

b) fomentar la formación y la investigación, e impulsar el intercambio de experiencia y conocimientos entre las instituciones y las asociaciones profesionales relacionadas con el tema;

c) alentar a las universidades y otras instituciones de investigación, públicas y privadas, a velar por la preservación de los datos relativos a las investigaciones.

Artículo 11 - Alianzas y cooperación

La preservación del patrimonio digital exige un esfuerzo constante por parte de gobiernos, creadores, editoriales, industriales del sector e instituciones que se ocupan del patrimonio.

Ante la actual "brecha digital" es necesario reforzar la cooperación y la solidaridad internacionales para que todos los países puedan garantizar la creación, difusión y preservación de su patrimonio digital, así como un acceso constante al mismo.

Se insta a los fabricantes, las editoriales y los medios de comunicación de masas a que promuevan y compartan sus conocimientos teóricos y técnicos.

El hecho de favorecer programas de educación y formación, acuerdos de aprovechamiento compartido de recursos y mecanismos de difusión de los resultados de investigaciones y prácticas idóneas democratizará el conocimiento de las técnicas de preservación de objetos digitales.

Artículo 12 - La función de la UNESCO

En virtud de su mandato y funciones, incumbe a la UNESCO:

a) incorporar los principios establecidos en esta Carta al funcionamiento de sus programas y promover su aplicación tanto dentro del sistema de las Naciones

Unidas como por las organizaciones internacionales, gubernamentales y no gubernamentales, relacionadas con la preservación del patrimonio digital;

b) ejercer de referente y de foro en el que los Estados Miembros, las organizaciones internacionales, gubernamentales y no gubernamentales, la sociedad civil y el sector privado puedan aunar esfuerzos para definir objetivos, políticas y proyectos que favorezcan la preservación del patrimonio digital;

c) impulsar la cooperación, sensibilización y creación de capacidades y proponer directrices éticas, jurídicas y técnicas normalizadas para apoyar la preservación del patrimonio digital;

d) basándose en la experiencia que adquirirá en los seis años venideros con la aplicación de la presente Carta y las directrices, determinar si se requieren nuevos instrumentos normativos para promover y preservar el patrimonio digital.

APÉNDICE B

Patente

Los desarrolladores de la PROGRAMMA 101 decidieron que debían proteger su producto frente a sus competidores, ya que sus partes podrían ser analizadas y copiadas. Por ello, en marzo de 1965 presentaron en Estados Unidos la patente número 3.495.222 [28], esta está firmada por Pier Giorgio Perotto y Giovanni De Sandre.

La patente que se puede ver en este apéndice, está constituida por varios esquemas eléctricos y estructurales, un resumen y una descripción del invento desde los puntos de vista del hardware y del software.

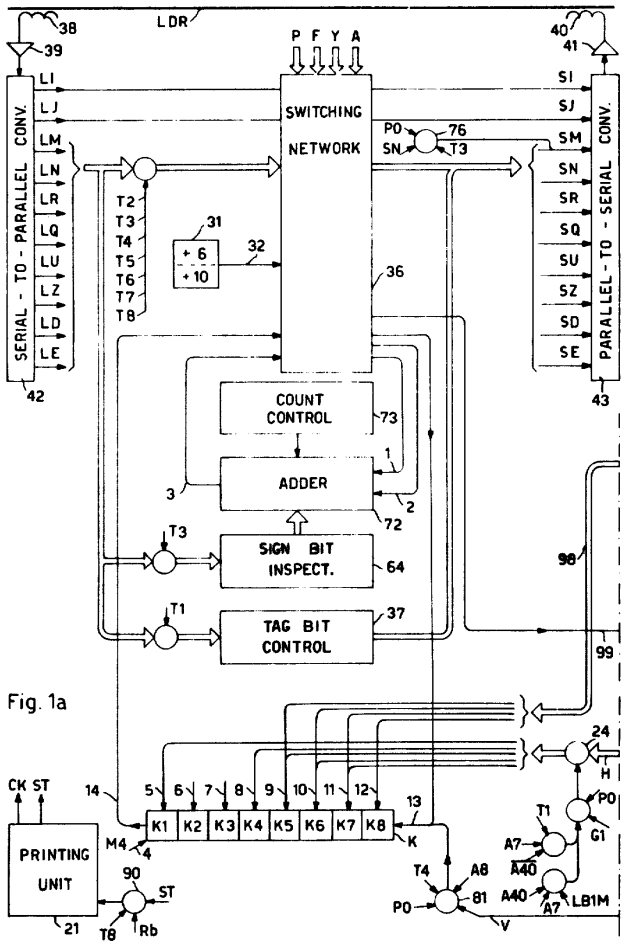


Fig. 1a

INVENTORS
PIER GIORGIO PEROTTO
GIOVANNI DE SANDRE
BY Scheinbaum and Mattare
ATTYS.

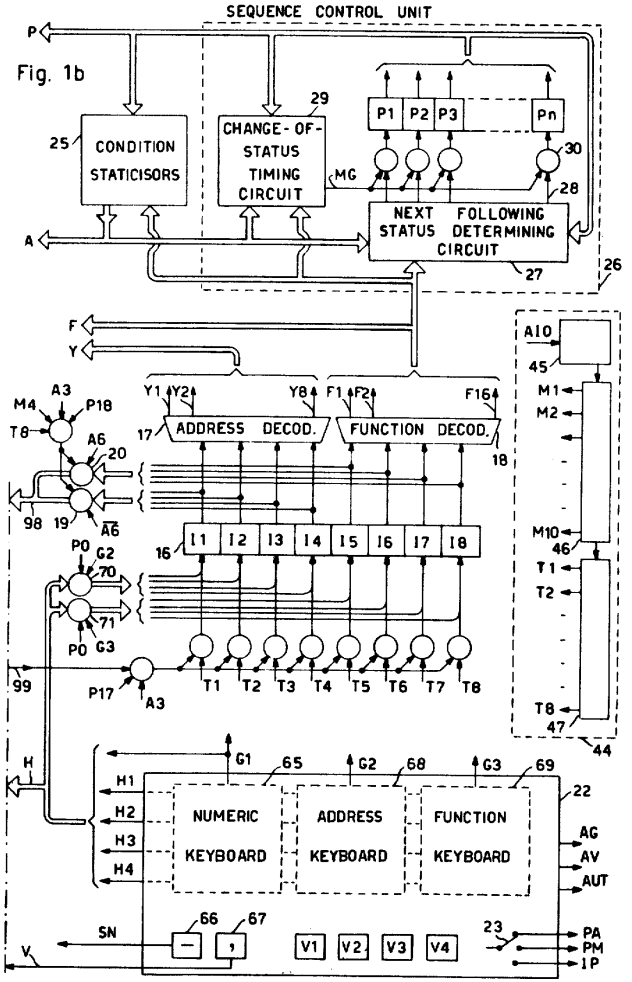


Fig. 1b

INVENTORS
PIER GIORGIO PEROTTO
GIOVANNI DE SANDRE
BY Scheinbaum and Mattare
ATTYS.

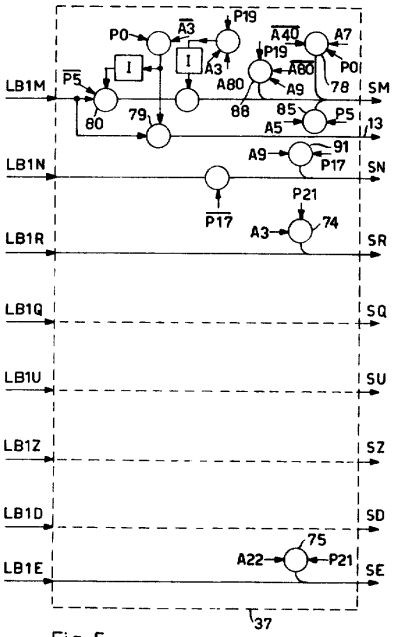


Fig. 5

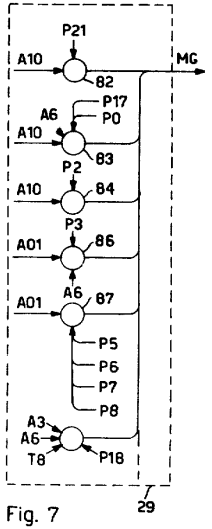


Fig. 7

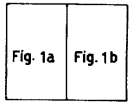


Fig. 2

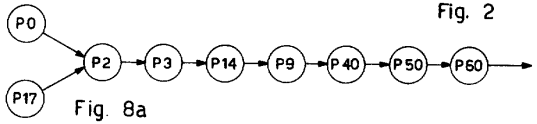


Fig. 8a

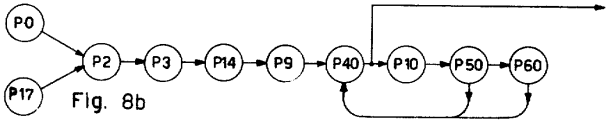


Fig. 8b

INVENTORS
PIER GIORGIO PEROTTO
BY GIOVANNI DE SANDRE
Schumacher and Mattare
ATTYS.

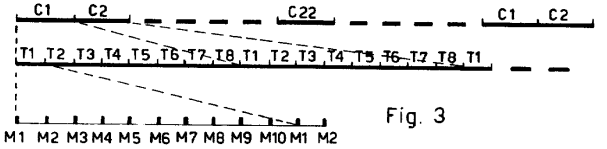


Fig. 3

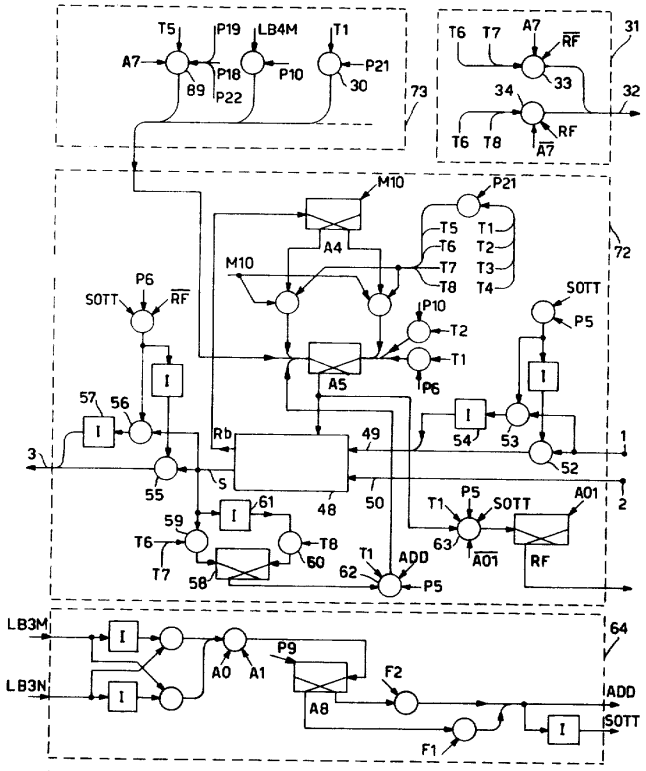
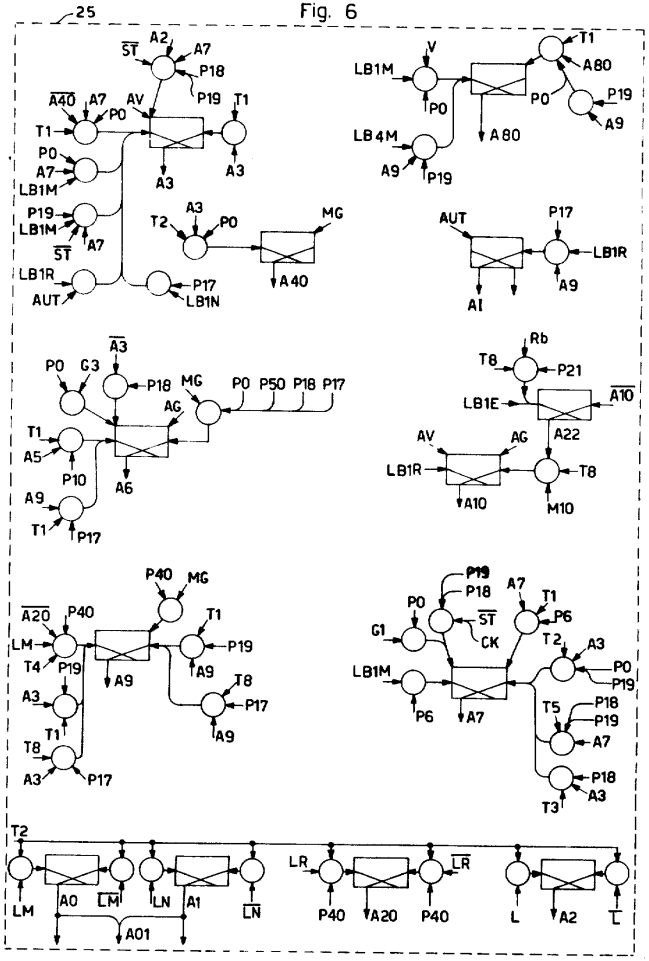


Fig. 4

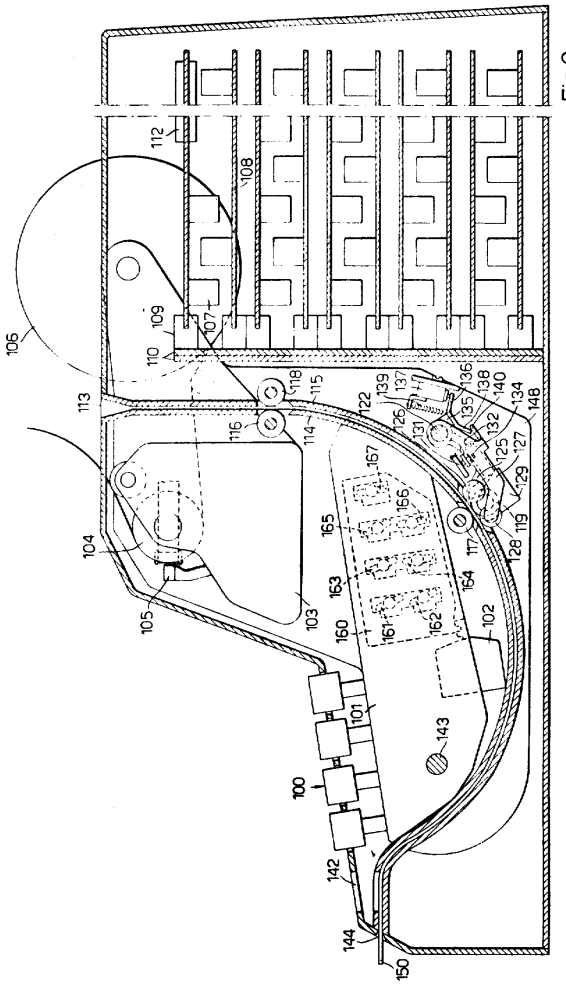
INVENTORS
PIER GIORGIO PEROTTO
BY GIOVANNI DE SANDRE
Schumacher and Mattare
ATTYS.

Fig. 6



INVENTORS
Pier Giorgio Perotto
BY
Giovanni De Sandre
Shoemaker and Mattare
ATTYS.

Fig.9



INVENTORS
Pier Giorgio Perotto
BY
Giovanni De Sandre
Shoemaker and Mattare
ATTYS.

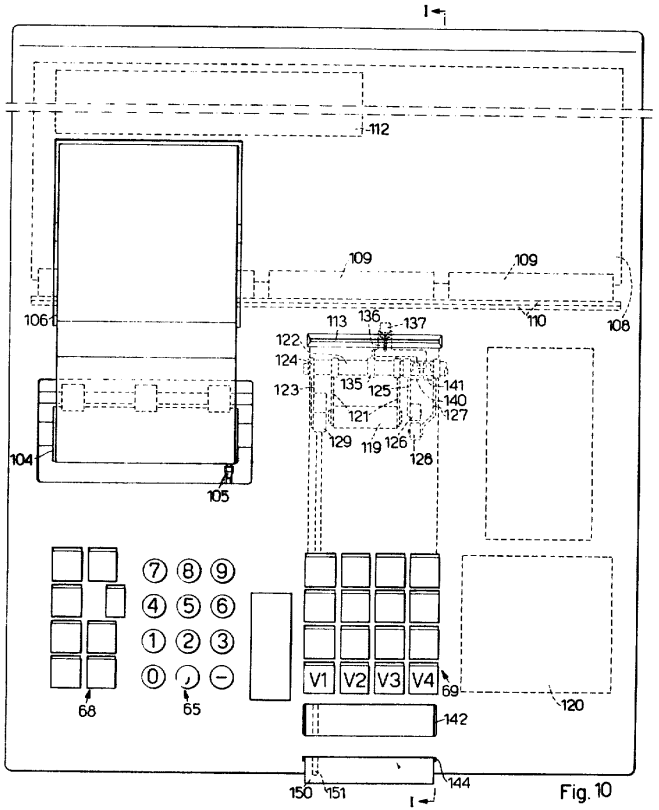
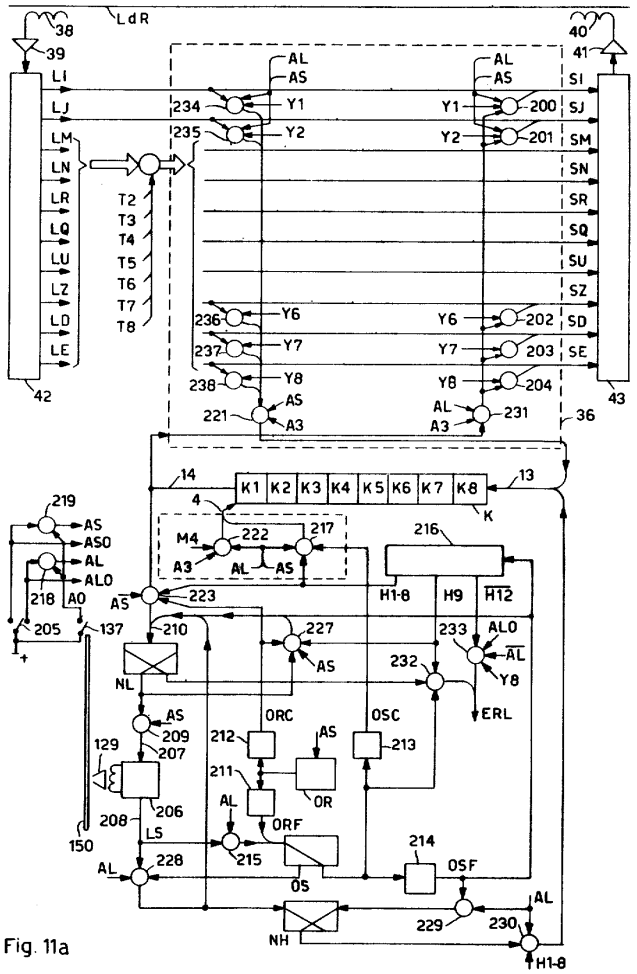


Fig. 10

INVENTORS
PIER GIORGIO PEROTTO
BY GIOVANNI DE SANDRE
Sherman and Mattare
ATTYS.



INVENTORS
PIER GIORGIO PEROTTO
BY GIOVANNI DE SANDRE
Sherman and Mattare
ATTYS.

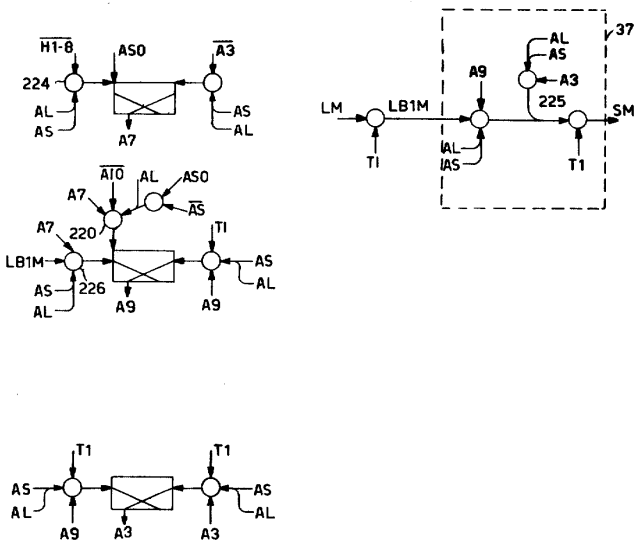


Fig. 11b

INVENTORS
Pier Giorgio Perotto
BY Giovanni De Sandre
Shoemaker and Mattare
ATTYS.

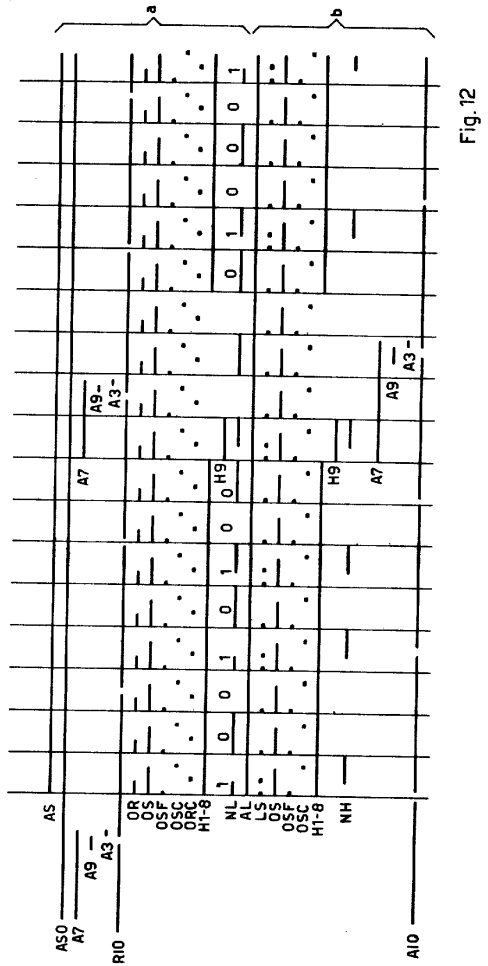


Fig. 12

INVENTORS
Pier Giorgio Perotto
BY Giovanni De Sandre
Shoemaker and Mattare
ATTYS.

United States Patent Office 3,495,222 Patented Feb. 10, 1970

3,495,222

PROGRAM CONTROLLED ELECTRONIC COMPUTER

Pier Giorgio Perotto, Turin, and Giovanni De Sandre, Sacile, Italy, assignors to Ing. C. Olivetti & C., S.p.A., Ivrea, Italy, a corporation of Italy

Continuation of application Ser. No. 435,828, Mar. 1, 1965. This application Jan. 12, 1968, Ser. No. 697,537
Claims priority, application Italy, Mar. 2, 1964, 4,933/64; Jan. 2, 1965, 469/65
Int. Cl. G11b 13/00; G06f 1/00
U.S. Cl. 340—172.5 16 Claims

ABSTRACT OF THE DISCLOSURE

In an electronic computer provided with a storage for storing a program comprising a series of instructions, with an instruction staticisor wherein a predetermined instruction is transferred from said program storage under the control of said program, and with executing means automatically operative upon entering said instruction into said instruction staticisor for executing said instruction, a set of control keys is manually operable for entering said instruction into said instruction staticisor, the actuation of said control keys automatically making operative said executing means out of the control of said program, and program record cards may be manually introduced in a record processing device comprised in said computer for making available said program in said program storage.

This application is a continuation of application Ser. No. 435,828, filed Mar. 1, 1965.

The present invention relates to a program controlled electronic computer, for instance a so called desk-top computer.

The known desk-top electronic computers are not adapted for being controlled by a program stored in their internal memory, whereby the number and complexity of the different operations they can perform is strictly limited. Therefore they are no more powerful in processing data than the mechanical desk-top calculators.

Some known medium-size computers have the ability to simulate a desk-top calculator under the control of a simulator program stored therein.

However, the structure of these computers is so complex that operation as mechanical calculators is uneconomical and difficult.

Moreover in the known program controlled computers the operator is given no sufficient control over the operation of the computer during the automatic execution of the program.

These and other disadvantages are obviated by the computer according to the invention, which is provided with a storage for storing a program comprising a series of instructions and with means controlled by said program for transferring a predetermined instruction from said program storage to an instruction staticisor and with means automatically operative upon entering said instruction into said instruction staticisor for executing said instruction and is characterized in that it comprises a set of control keys for entering said instruction into said instruction staticisor, whereby the manual actuation of said control keys automatically makes said executing means operative out of the control of said program.

Therefore the computer may operate either automatically under the control of the stored program or manually under the control of the keyboard.

According to another feature of the invention, the computer, which is provided with a record processing device for reading records bearing said program and entering the

read program into said program storage, is characterized in that said records are in form of cards each one containing a program, said record processing device being so associated with said program storage that by manually introducing a single record in said record processing device said program is made available in said program storage for controlling said computer.

According to another feature of the invention, the computer is provided with subroutine keys, manually operable to cause the computer to execute any selected subroutine of the program stored in the memory, whereby automatic execution of preselected subroutines may be included in the manual operation.

This feature, in connection with a novel association of said subroutine keys with said record processing device, gives the computer a great flexibility as if it were provided wtih an unlimited number of control keys.

Other objects, features and advantages of the invention will be apparent from the following description, made by way of example and not in a limiting sense, in connection with the accompanying drawings, wherein:

FIGS. 1*a* and 1*b* show a block diagram of the circuits of the computer according to an embodiment of the invention;

FIG. 2 shows how FIGS. 1*a* and 1*b* are to be composed;

FIG. 3 shows a time diagram of some clock signals of the computer according to FIGS. 1*a* and 1*b*;

FIG. 4 shows an adder used in an embodiment of the computer according to the invention;

FIG. 5 shows a circuit for controlling the tag bits used in the computer according to the invention;

FIG. 6 shows a group of bistable devices of the computer according to FIGS. 1*a* and 1*b*;

FIG. 7 partially shows a circuit for timing the switching from a status to the next following status in the computer according to the invention;

FIG. 8*a* is a diagram showing the sequence of statuses of the computer in the addition or subtraction according to an embodiment of the invention;

FIG. 8*b* is a diagram showing the sequence of statuses of the computer in the multiplication or division according an embodiment of the invention;

FIG. 9 shows a vertical section of an embodiment of the computer according to the invention;

FIG. 10 shows a top view of the computer of FIG. 9;

FIGS. 11*a* and 11*b* show some circuits of the computer involved in the card reading and writing operation;

FIG. 12 shows a time diagram of the card reading and recording operation.

General description

The computer comprises a storage made of a magnetostrictive delay line LDR including for instance ten registers I, J, M, N, R, Q, U, Z, D, E and provided with a reading transducer **38** feeding a reading amplifier **39** and with a writing transducer **40** fed by a writing amplifier **41**.

Each memory register comprises for instance 22 decimal denominations, each one comprising eight binary denominations, whereby each register may store up to 22 eight-bit characters. Both the characters and the bits are processed in series. Therefore a train of 10·8·22 binary signals recirculates in the delay line LDR.

The ten first occurring binary signals represent the first bit of the first decimal denomination of the register R, N, M, J, I, Q, U, Z, D and E respectively, the ten next following binary signals represent the second bit of said first decimal denomination of said registers respectively, etc.

Assuming for instance said binary signals are recorded

in the delay line so as to be spaced 1 microsecond from each other, the signals belonging to a certain register will be spaced 10 microseconds from each other. Otherwise stated, each register comprises a train of 8·22 binary signals spaced 10 microseconds from each other, the trains belonging to the several registers being diseplaced 1 microsecond from each other.

The reading amplifier **39** feeds a serial-to-parallel converter **42**, which produces over ten separate outputs lines LR, LM, LN, LJ, LI, LE, LD, LQ, LU and LZ, ten simultaneous signals representing the ten bits stored in the same binary denomination of the same decimal denomination of the same decimal denomination of the ten registers respectively.

Therefore, at a given instant ten signals representing the first bit of the first decimal denomination of the ten registers are simultaneously present on said ten output lines; ten microseconds later, ten signals representing the second bit of the first decimal denomination are present on said output lines, etc.

Each group of ten signals simultaneously delivered on the output lines of the converter **42** after being processed is fed to a parallel-to-serial converter **43**, which feeds the writing amplifier **41** with said ten signals restored in their previous serial order and spaced 1 microsecond from each other, whereby the transducer **40** writes in the delay lines said signals either unchanged or modified according to the operation of the computer, while maintaining their previous relative location. Therefore it is apparent that the single delay line LDR is equivalent, with respect to the external circuits which process its contents, to a group of ten delay lines working in parallel, each one containing a single register and provided with an output line LR, LM, LN, LJ, LI, LE, LD, LQ, LU and LZ respectively and with an input line SR, SM, SN, SJ, SI, SE, SD, SQ, SU and SZ respectively.

This interleaved arrangement of the signals in the delay line allows all the registers of the computer to be contained in a single delay line provided with a single reading transducer and a single writing transducer, whereby the ultimate cost of the memory does not exceed the cost of a delay line containing only one register. Moreover, as the pulse repetition frequency in the delay line is ten times greater than in the other circuits of the computer, it is possible to simultaneously attain a good uilization of the storage capacity of the delay line while using low speed switching circuits in the other parts of the computer, thus substantially reducing the cost of the machine.

As the delay line storage is cyclic in nature, the operation of the computer is divided into successive memory cycles, each cycle comprising twenty-two digit periods C1 to C22, and each digit period being divided into eight bit periods T1 to T8.

A clock pulse generator **44** produces on the output lines T1 to T8 successive clock pulse, each one having a duration which indicates a corresponding bit period, as shown in the time diagram of FIG. 3. Otherwise stated, the output terminal T1 is energized during the entire first bit period of each one of the twenty-two digit periods, the output terminals T2 is similarly energized during the entire second bit period of each one of the twenty-two digit periods, etc.

The clock pulse generator **44** is synchronized with the delay line LDR, as will be seen, in such a way that the beginning of the n^{th} generic bit period of the m^{th} generic digit period coincides with the instant in which the ten binary signals representing the ten bits read in the n^{th} binary denomination of the m^{th} decimal denomination of the ten memory registers begin to be available on the outputs lines of the serial-to-parallel converter **42**. Said binary signals are staticized in the converter **42** for the entire duration of the corresponding bit period. During the same bit period the signals representing the ten bits produced by processing said ten bits read out of the delay

line LDR are fed to the parallel-to-serial converter **43** and written in the delay line.

More particularly the generator **44** produces during each bit period ten pulses M1 to M10 (FIG. 3). The pulse M1 defines the reading time, that is the instant when the serial-to-parallel converter **42** begins to make available the bits pertaining to the present bit period, whereas the pulse M4 indicates the writing time, that is the instant when the processed bits are fed to the parallel to-serial converter **43** for being written into the delay line LDR.

The generator **44** comprises an oscillator **45** which, when operative, feeds a pulse distributor **46** with pulses having the frequency of said pulses M1 to M10, a frequency divider **47** fed by said distributor being arranged to produce the clock pulses T1 to T8.

The oscillator **45** is operative only as long as a bistable device A10 (FIG. 6) remains energized, said bistable device being controlled by signals circulating in the delay line LDR, as will be seen.

Each decimal denomination of the memory LDR may contain either a decimal digit or an instruction. More particularly the registers I and J, which are designated as first and second instruction register respectively, are adapted to store a program comprising a sequence of **44** instructions written in the 22 decimal denominations of the registers I and J respectively.

The remaining registers M, N, R, Z, U, Q, D, E are normally numerical registers, each one adapted to store a number having a maximum length of 22 decimal digits.

Each instruction is made of eight bits B1 to B8 stored in the binary denominations T1 to T8 respectively of a certain decimal denomination: the bits B5 to B8 represent one out of 16 operations F1 to F16 whereas bits B1 to B4 generally represent the address of an operand upon which said operation is to be performed.

Each decimal digit is represented in the computer by means of four bits B5, B6, B7, B8 according to a binary-coded decimal code. In the delay line memory LDR said four bits are recorded in the last occurring four binary denominations T5, T6, T7, T8 respectively of a certain decimal denomination, while the remaining four binary denominations are used to store certain tag bits. More particularly, in this decimal denomination the binary denomination T4 is used for storing a decimal-point bit B4, which is equal to "0" for all the digit of a decimal number except the first entire digit after the decimal point. The binary denomination T3 is used for storing a sign bit B3, which is equal to "0" for all the decimal digits of a positive number and equal to "1" for all the decimal digits of a negative number. The binary denomination T2 is used for storing a digit-identifying bit B2, which is equal to "1" in each decimal denomination occupied by a decimal digit of a number and equal to "0" in each unoccupied decimal denomination (non significant zero).

Therefore the complete representation of a decimal digit in the memory LDR requires the seven binary denominations T2, T3, T4, T5, T6, T7 and T8 of a given decimal denomination.

The remaining binary denomination T1 is used for storing a tag bit B1 whose meaning is not necessarily related to the decimal digit stored in said denomination.

In the following description a bit stored in a binary denomination *a* of a certain decimal denomination of a register *b* will be designated as B*ab*, and the signal obtained when reading said bit out of the delay line will be designated LB*ab*.

A bit B1R="1" stored in the first decimal denomination C1 of the register R is used to start the clock pulse generator **44** at the beginning of each memory cycle; a bit B1E="1" stored in the 22^{nd} decimal denomination C22 of the register E is used to stop the generator **44**; a bit B1N="1" stored in the n^{th} decimal denomination of

the register N indicates that during the execution of a program the next following instruction to be executed is the instruction stored in said n^{th} decimal denomination of the register I or J; a bit B1M="1" stored in the n^{th} decimal denomination of the register M indicates: when introducing a number from the keyboard into the register M, that the decimal digit next introduced is to be stored in the $(n-1^{st})$ decimal denomination; when introducing an instruction from the keyboard, that the next following instruction is to be stored in the n^{th} decimal denomination of the register I or J; when printing a number stored in any register selected among the registers of the delay-line, that the next following digit to be printed is the digit stored in the n^{th} decimal denomination of said register; when adding together two numbers, that the digit of the sum stored in the n^{th} decimal denomination of the register N shall be thereafter corrected by adding a filler digit thereto, as will be seen; a bit B1U="1" stored in the n^{th} decimal denomination of the register U indicates that the execution of a main program routine has been interrupted at the n^{th} instruction of the register I or J for beginning the execution of a subroutine. Therefore the tag bits B1R, B1E are used to represent fixed reference points in the various registers (beginning and end respectively); the tag bits B1N, B1M and B1U represent movable reference points within the registers; moreover the bits B1M are used, when performing an addition, to record, for each decimal denomination, an information pertaining to an operation performed or to be performed upon said denomination.

The regeneration and the modification and shifting of said tag bits **B1** are preformed by a tag-bit control circuit **37**.

The computer comprises also a binary adder **72** provided with a pair of input lines **1** and **2** for concurrently receiving two bits to be added to simultaneously produce on the output line **3** the sum bit. More particularly, in a first embodiment shown in FIG. 4, the adder comprises a binary addition network **48**, adapted to provide on the output lines S and R*b* the binary sum and the binary carry, respectively, produced by summing up two bits concurrently fed to the input lines **49** and **50** respectively and the previous binary carry bit resulting from the addition of the next preceding pair of bits, said previous binary carry bit being staticized in a carry bit storage **A5** made of a bistable circuit. The signals representing the two bits to be added last from the pulse **M1** to the pulse **M10** of the corresponding bit period, and the signals representing the sum bit S and the carry bit R*b* are substantially simultaneous thereto. The previous carry bit is stored in the bistable circuit **A5** from the pulse **M10** of the next preceding bit period until the pulse **M10** of the present bit period.

The new carry bit R*b* is transferred to a bistable circuit **A4**, in which it is staticized until the pulse **M10** causes said new carry bit to be transferred into the bistable circuit **A5**, where it is staticized during the entire next following bit period so as to feed in proper time the addition network **48** during the addition of the next following pair of bits.

The input line **1** of the adder may be connected to the input line **49** of the addition network **48** either directly via a gate **52** or through an inverter **54** via a gate **53**. Therefore it is apparent that in the first case each decimal digit is introduced without modification into the adder, whereas in the second case, as said digit is represented in binary code, the complement of said digit to 15 is introduced in the adder.

The gates **52** and **53** are controlled by a signal SOTT produced by a sign-bit processing circuit which will be described later.

The output line S of the addition network **48** may be connected to the output line **3** of the adder either directly via a gate **55** or via a gate **56** and an inverter **57** acting to complement the decimal digits to 15.

A bistable device **58** is energized through a gate **59** by every bit equal to "1" appearing on the output line S of the addition network **48** during the bit periods T**6** and T**7**, and is deenergized through an inverter **61** and a gate **60** by every bit equal to "0" appearing on said output line S during the bit period T**8**.

Therefore, upon completion of the addition of a pair of decimal digits during the n^{th} generic digit period, the circumstances that the bistable device **58** remains energized after the last bit period T**8** of said digit period indicates that the sum digit is greater than nine and less than sixteen, whereby a decimal carry is to be transmitted to the next following decimal denomination. Through a gate **62** the output signal of the bistable device **58** indicating the presence of said decimal carry is fed into the carry storage **A5**, which is adapted to enter said decimal carry into the adding network **48** in the next following digit period $C(n+1)$.

A decimal carry toward said next following decimal denomination is to be transmitted also in the case during said bit period T**8** of the present digit period Cn a binary carry R*b***8** is produced by summing up the two most significant bits **B8**. Since this binary carry indicates that the sum digit is greater than fifteen. The transmission of the decimal carry is made in this case by the bistable devices **A4** and **A5** in the manner described above.

Therefore in all cases the circumstance that the bistable device **A5** is energized after the last bit period **T8** of said digit period Cn means that there is a decimal carry to be transmitted from said digit period Cn to the next following digit period $C(n+1)$.

Should said digit period Cn be the digt period in which the last (most significant) decimal digit among the digits of the two numbers to be added occurs, then through a gate **63** said decimal carry is stored into a bistable device RF. Therefore the bistable device RF when energized indictates that there exists an end carry resulting from the addition of the two most significant decimal digits.

Moreover the computer is provided with a shift register K (FIG. 1*a*) comprising eight binary stages **K1** to **K8**. Upon receiving a shift pulse over a terminal **4**, the bits stored in the stages **K2** and **K8** are shifted into the stages **K1** to **K7** respectively, while the bits which are then present on the input lines **5, 6, 7, 8, 9, 10, 11, 12, 13** are transferred into the stages **K1, K2, K3, K4, K5, K6, K7, K8** and again **K8** respectively.

The pulses **M4** produced by the pulse distributor **46** (FIG. 1*b*) are used as shift pulses for the register K, which therefore receives one shift pulse during each bit period, that is eight shift pulses during each digit period. The contents of each stage of the register K remains unchanged from the pulse **M4** of each bit period until the pulse **M4** of the next following bit period. Therefore it is apparent that a bit fed to the input line **13** of the register K during a certain bit period will be available on the output line **14** of the register K after eight bit periods, that is one digit period later, whereby under these conditions the register K acts as a section of delay line having a length corresponding to one digit period.

By connecting whatsoever memory register X and the shift register K in a closed loop while leaving all the remaining registers with their outputs directly connected to their respective inputs to form a closed loop, said register X is effectively lengthened one digit period with respect to said remaining registers. In this lengthened register **X**, the denomination which is read from the delay line concurrently with the n^{th} decimal denomination of the remaining memory registers, that is during the n^{th} digit period since the reading of the bit **B1R** which starts the generator **44**, is conventionally defined as the n^{th} decimal denomination. Therefore during each memory cycle the contents of the register X will be shifted one decimal denomination, that is delayed one digit period, with respect to the other registers.

Moreover the register K, due to its ability to acts as a

delay line, may be used as a counter according to the principles shown at page 198 of the book "Arithmetic Operation in Digital Computers," by R. K. Richards, 1955. More particularly, when its output line **13** and its input line **14** are connected to the output line **3** and to the input line **1** of the adder **72**, respectively, while the input line **2** of the adder receives no signal, said counter is adapted to count successive counting pulses which are fed to the carry storing bistable device **A5** according to the following criterion. By considering the eight bits contained in the register K as a binary number comprising eight binary denominations, a counting pulse may be fed into the bistable circuit **A5** whenever the less significant binary denomination is read out of the register K over the output line **14**. Therefore the counting pulses shell be spaced in time one digit period or a multiple thereof.

The register K is also adapted to act as a buffer memory for temporarily storing a decimal digit or the address part of an instruction or the function part of an instruction to be printed by a printing unit **21** (FIG. 1*a*).

The register K is also adapted to act as a parallel-to-serial converter when transferring data or instruction from the keyboard **22** (FIG. 1*b*) into the delay line memory LDR.

The computer comprises also an instruction staticisor **16** including eight binary stages **I1** to **I8** for storing the eight bits **B1** to **B8** of an instruction respecitvely.

The first four stages **I1** to **I4** containing the address bits **B1** to **B4** of said instruction feed an address decoder **17** having eight output lines **Y1** to **Y8**, each one corresponding to one of the eight addressable memory registers, and being energized when the combination of said four bits represents the address of said register. The address of the register M is represented by four bits equal to "0," whereby the register M is automatically addressed when no address is explicitly given. The remaining four stages **15** to **18** containing the function bits **B5** to **B8** of said instruction feed a function decoder **18** having a set of outputs **F1** to **F16**, each output being energized when the combination of said bits **B5** to **B8** represents a corresponding function.

Moreover the outputs of the stages **I1** to **I4** and the output lines of the stages **I5** to **I8** may be connected, via gates **19** and **20** respectively, to the input lines of the stages **K5** to **K8** of the register K respectively in order to print out the address and the function respectively staticized in said stages.

A switching network **36** (FIG. 1*a*) is provided for selectively interconnecting according to various patterns hereinafter specified, the ten memory registers, the adder **72**, the shift register K and the instruction staticisor **16** in order to properly control the transmission of data and instructions to and from the various parts of the computer. Switching network **36** is made of a diode matrix or transistor NOR-circuit matrix or equivalent switching means having no storage properties.

The selection of the memory registers according to the present address indicated by the decoder **17** is also performed by the switching network **36**.

The keyboard **22** for entering the data and the instructions and for controlling the various functions of the computer comprises a numeric keyboard **65** including ten numeral keys **0** to **9** which serve the purpose of entering number into the memory register M via the buffer register K, in a preferred embodiment the register M being the only memory register accessible from the numeral keyboard. Moreover the keyboard **22** comprises an address keyboard **68** provided with keys each one controlling the selection of a corresponding register of the delay line memory LDR.

The keyboard **22** comprises also a function keyboard **69**, including keys each one corresponding to the function part of one of the instructions the computer can execute.

The three keyboards **65, 68** and **69** control a mechanical decoder made of code bars cooperating with electrical switches for producing on four lines **H1, H2, H3, H4** four binary signals representing either the four bits of a decimal digit set up on the keyboard **65** or the four bits of an address set up on the keyboard **68**, or the four bits of a function set up on the keyboard **69**, said decoder being also adapted to energize either an output line **G1** or **G2** or **G3** to indicate whether the keyboard **65** or **68** or **69** respectively has been operated.

A decimal point key **67** and a negative algebraic sign key **66**, when operated, directly produce a binary signal on the line V and SN respectively.

Some instructions the present computer can execute are listed below, the letter Y designating the selected register corresponding to the address staticized in the staticisor **16**:

(F1) Addition: transfer the number stored in the selected register Y into the register M, then add the contents of the register M to the contents of the register N and store the result in the register N, that is symbolically: $Y--M$; $(N+M)--N$;
(F2) Subtraction: similarly $Y--M$; $(N-M)--N$;
(F3) Multiplication: $Y--M$; $(N\cdot M)--N$;
(F4) Division: $Y--M$; $(N:M)--N$;
(F5) Transfer from M: transfer the contents of the register M into the selected register, that is $M--Y$;
(F6) Transfer into N: transfer into the register N the contents of the selected register, that is $Y--N$;
(F7) Exchange: transfer the contents of the selected register into the register N and vice versa, that is $Y--N$; $N--Y$;
(F8) Print: print-out the contents of the selected register Y;
(F9) Print and zeroizes: print-out the contents of the selected register Y and zeorize same;
(F10) Program stop: stop the automatic execution of the program and wait until operator enters a datum into the keyboard; introduce said datum into the selected register Y (thereafter either automatic program execution or manual operation may be continued);
(F11) Extract from the register I one out of the first eight characters as specified by the address contained in the present instruction, and transfer said character into register M;
(F12) Jump to the program instruction specified in the present instruction, unconditional;
(F13) Jump, conditional.

The computer may be selectively preset to operate according to three modes, namely "manual," "automatic" and "entering program" depending on whether a three-position commutator **23** generates a signal PM, PA or IP respectively. All the aforementioned instructions may be executed in the automatic operation; the first nine instructions may also be executed in the manual operation.

During the program entering operation, the signal IP being present, the address keyboard **68** and the function keyboard **69** are operable to enter the program instructions into the registers I and J via the buffer register K. For this purpose the outputs **H1** and **H4** of the keyboard decoder may be connected, via gate **24**, to the inputs **8** to **11** respectively of the register K. In the meantime, the keyboard **65** is inoperative.

During the automatic operation, in which the program previously entered into the memory LDR is executed, the address keyboard and the function keyboard are inoperative.

The automatic operation comprises a sequence of instruction-extract phases and instruction-execute phases. More particularly during an extract phase an instruction is extracted from the program register I, J and transferred into the staticisor **16**; this phase is automatically followed by an execution phase, in which the computer under the control of said staticized instruction executes said instruction; this execution phase is automatically followed by an extraction phase for the next following instruction,

which is then extracted and staticized in lieu of the preceding one etc. As long as an instruction is staticized in the staticisor **16**, the numeric register indicated by the address part of said instruction remains continuously selected, and the decoder **18** continuously produces the function signal corresponding to the function part of said instruction. During the automatic operation, also the numeric keyboard is normally inoperative, because the computer operates upon the data previously entered into the memory. This keyboard is operated only when the program instruction at present staticized is the stop instruction **F10**. It is apparent that this instruction allows much more data to be processed than the computer memory may contain.

During the manual operation the numeric keyboard, the address keyboard and the function keyboard may be all operative. More particularly according to this mode of operation the address keyboard and the function keyboard may be caused by the operator to cause the computer to perform a sequence of operations similar to any sequence performed during the automatic operation. For this purpose the operator enters via the keyboard an address and a function, which are therefore staticized via gates **70** and **71** respectively in the staticisor **16** just like during an instruction-extract phase in the automatic operation. Moreover, by entering said instruction (address and function) into the keyboard, an instruction-execution phase is automatically instituted for executing said entered instruction in a manner similar to the execution phase in the automatic operation. Upon completion of said instruction-execution phase the computer stops and waits for a new instruction entered by the operator through the keyboard.

As previously mentioned, when no address key is operated, the register M, which is specialized to receive the data from the keyboard, is automatically addressed. Therefore, when entering via the keyboard one of the instructions **F1**, **F2**, **F3**, **F4** corresponding to the four fundamental arithmetic operations, the operator may select not to operate the address keyboard but instead to enter a number through the numeric keyboard; in this case said operation will be performed upon said entered number. Therefore during the manual operation any arithmetic operation corresponding to the key depressed in the function keyboard **69** may be performed either upon a number previously entered into the register M via the numeric keyboard **65** or upon a number stored in a memory register selected by means of the address keyboard.

Moreover it has been seen that during the automatic operation the functions specified in the instructions are executed upon the data previously entered in the memory. Before pushing the button AUT to start the automatic program execution, the operator after having set the computer to operate in the manual mode, may enter each one of said initial data, by first entering said datum through the numeric keyboard into the register M, then depressing the address key corresponding to the register in which said datum is to be stored, and then depressing the function key corresponding to the transfer instruction **F5**.

The computer comprises also a group of bistable devices collectively represented by a box **25** in FIG. 1*b* and in more details in FIG. 6. These bistable devices are used, inter alia, to staticize some internal conditions of the computer, the output signals of said bistable devices representing said conditions being collectively designated by the reference letter A in the block diagram of FIG. 1.

More particularly, the bistable device **A0** is energized during each memory cycle upon reading in the register M the first binary denomination **T2** storing a digit indicating bit **B2** equal to "1" and is thereafter deenergized upon reading the first binary denomination **T2** storing a digit indicating bit **B2** equal to "0," whereby the bistable device **A0** remains energized during the entire time interval spent in reading out the number stored in the register M. Otherwise stated, the bistable device **A0** indicates within each memory cycle the length and the position of the number stored in the register M. It is to be pointed out that according to a feature of the present invention said length and said position are completely variable.

The bistable devices **A1** and **A2** are adapted to give a similar indication as to the length and position of the number stored in the register N and Y respectively, Y designating the register at present addressed and selected. For this purpose the bistable devices **A1** and **A2** are controlled by the output LN of the register N and by the output L of the selected register Y respectively. The outputs of the bistable devices **A0** and **A1** are combined to produce a signal **A01** which lasts, during each memory cycle, from the reading time of the first decimal digit among the decimal digits of the numbers M and N until the reading time of the last occurring decimal digit among said decimal digits.

The bistable device **A3** is normally used to distinctively indicate a certain digit period during which a certain operation is to be performed, said indication being obtained in that it remains energized during said digit period and deenergized during the other digit periods.

The bistable device **A7** is normally used to distinctively indicate a certain memory cycle or a part thereof during the operation of the input and output units of the computer.

The bistable devices **A6**, **A8**, **A9** are used to indicate the occurrence of certain conditions during the execution of certain instructions.

The function of other bistable devices of the group **25** will be described later.

The computer is also provided with a sequence control unit **26** comprising a group of status-indicating bistable devices **P1** to *Pn*, which are energized one at a time, whereby at any time the computer is in a certain status corresponding to one of the bistable devices **P1** to **P***n* at present energized. In its operation the computer goes through a sequence of statuses, and accomplishes certain elemental operations during each status. The sequence of said statuses is determined according to a criterion established by a logical network **27**. More particularly on the basis of the present status of the computer indicated by the bistable devices **P1** to **P***n* via the line P, of the instruction at present staticized in the staticisor **16** and indicated by the decoder **18** via the line F, and of the present internal conditions of the computer indicated by the group of condition-staticizing bistable devices **25** via the line A, said network **27** decides what status must follow and gives an indication of said decision by energizing the output **28** which corresponds to said status. Thereafter a timing network **29** produces a change-of-status timing pulse MG, whereby one of the bistable devices **P1** to **P***n* corresponding to said next following status is energized via the gate **30** corresponding to said output **28**, while all the remaining status-indicating bistable devices of the group **P1** to **P***n* are deenergized.

Entering a number into the memory via the keyboard

The status **P21** is followed by the status **P0** wherein the data may be entered into the memory via the keyboard.

In the status **P0** the switching network **36** permanently connects the memory register M and the shift register K to build up a closed loop, whereby the register M is lengthened one digit period. In the meantime all the remaining registers have their output directly connected to their respective input so as to build up a closed loop, whereby their contents is continuously regenerated so as to rematin unchanged during the following memory cycles. Also the tag bits **B1** of said remaining registers are continuously regenerated through the control circuit **37**, whereby the entire contents of all the registers but the register M remains unchanged during said status **P0**.

The timing signal MG which causes the computer to

switch from the status **21** to the status **P0** resets the bistable device **A40**. The operator pushes either the minus sign key **66** or no key depending on whether the number to be entered is negative or positive. In the first case the signal SN produced by the pushed key causes a negative sign bit B3="1" to be written via a gate **76** in the third binary denomination of all the decimal denominations of the register M. Thereafter the operator pushes the numeric key corresponding to the first decimal digit to be entered. Therefore the electrical contacts associated with the keyboard **22** produce the four binary signals **H1**, **H2**, **H3**, **H4** representing said decimal digit and a signal **G1** indicating that said four signals pertain to a numeric character entered via the numeric keyboard **65**. The duration of all said signal produced by the keyboard is more than one memory cycle.

The beginning (leading edge) of said signal **G1** energizes the bistable device **A7**. At a certain instant which may occur either before or after said leading edge, the synchronizing bit **B1R** circulating in the delay line starts the generator **44**. During the first clock pulse **T1** produced by the generator **44** after the energization of the bistable device **A7**, the pulse **M4** by opening the gate **24** causes the bits **H1**, **H2**, **H3**, **H4** and **G1**, to be transferred from the keyboard **22** into the stages **K4**, **K5**, **K6**, **K7** and **K1** of the register K respectively. Since the depressing of the key in the keyboard **22** is not synchronized with the generator **44**, said first clock pulse T1 may coincide with the first bit period of whatsoever digit period $C(n+1)$ among the twenty-two digit periods of the present memory cycle. Therefore at the beginning of said clock pulse **T1** the stages **K1** to **K8** of the register K will contain the binary denominations **B1** to **B8** respectively of the n^{th} decimal denomination of the register M. At the pulse **M4** of said bit period **T1** the bits of the binary denominations **B2** to **B8** of said n^{th} decimal denomination and the bit of the first binary denomination **B1** of the next following decimal denomination $C(n+1)$ will be transferred into the stages **K1** to **K8** of the register K respectively. At the same pulse **M4** the bits **H1**, **H2**, **H3**, **H4** and **G1** are entered from the keyboard **22** into the register K. Therefore these bits are written into the binary denominations **B5**, **B6**, **B7**, **B8** and **B2** respectively of said n^{th} decimal denomination C*n* of the register M, the four first-mentioned bits representing the entered digit and the fifth bit being a digit-indicating bit. As previously explained, the binary denomination **B3** has already been occupied by a sign bit.

Therefore it is apparent that the first digit entered via the keyboard is written at random in a certain n^{th} decimal denomination, which is the first decimal denomination first reaching the reading and writing transducers **38** and **40** after operation of the corresponding key.

Moreover at said pulse **M4** of said first bit period **T1** of the digit period $C(n+1)$ the output SM of the tag-bit controlling circuit **37** is energized because the output of the gate **78** is energized. Therefore a tag bit B1M="1" is written in the first binary denomination of said n^{th} decimal denomination of the register M, just ahead of the digit being introduced from the keyboard. Moreover said clock pulse **T1** energizes the bistable device **A3**, which is thereafter deenergized by the next following pulse **T1**, thus remaining energized only during said $(n+1st)$ digit period in order to designate the digit period during which the digit set up on the keyboard is entered in the register M.

The clock pulse **T2** of said digit period $C(n+1)$ deenergizes the bistable device **A7**, to inhibit said digit from being entered once more in the register M in the next following cycle, whereby said digit is entered only once in the register M, despite the fact that the corresponding key is held depressed during more than one memory cycle. It is thus apparent that the function of the bistable device **A7** in this case is to distinguish the first memory cycle from the following memory cycles when entering a digit via the keyboard. Moreover the same clock pulse **T2** energizes the bistable device **A40**, which will thus remain energized also during the setting up of the following digits on the keyboard in order to distinguish the first set up digit from the following ones. This is because the first entered digit is written at random in a decimal denomination of the register M, whereas the following digits must be written in the successive decimal denominations of the register M according to an ordered sequence. The purpose of the bistable device **A40** is to determine this difference in the digit entering operation. Said first entered digit circulates during the following memory cycles in the register M and in the register K which are connected into a closed loop as previously explained. In the tag-bit controlling circuit **37** also the tag bits **B1M** are caused to be stepped through the shift register K because they are transferred from the output LM of the register M to the input **13** of the register K since gate **79** instead of gate **80** is opened, whereby said bit B1M="1" remains recorded in the n^{th} decimal denomination occupied by said first entered digit, while the tag bit recorded in the first binary denomination of the remaining decimal denominations of the register M continues to be B1M="0."

Thereafter the second decimal digit of the number to be entered is set up on the keyboard, which therefore produces the binary signals **H1**, **H2**, **H3**, **H4** representing said digit and the signal **G1**. As previously stated, these signals have a duration corresponding to more than one memory cycle.

As in the case of the first entered digit, the beginning of the signal **G1** energizes the bistable device **A7**. Upon reading the tag bit B1M="1" recorded in the n^{th} decimal denomination of the register M, that is the denomination occupied by the first entered digit, the bistable device **A3** is energized. The bistable device **A3** will be thereafter denergized by the next following clock pulse **T1**, whereby it remains energized only during the n^{th} digit period, which begins when said tag bit B1M="1" is read from the delay line LDR. It is to be pointed out that when reading said bit B1M="1" located at the beginning of the n^{th} decimal denomination of the register M, the $(n-1^{st})$ decimal denomination is in the register K, while the $(n-2^{nd})$ decimal denomination, having just been rewritten in the register M, is at the beginning of the delay line.

When reading said tag bit **B1M**, the pulse **M4** by opening the gate **24** causes the binary signals **H1**, **H2**, **H3**, **H4** and **G1** to be transferred from the numeric keyboard **65** into the stages **K4**, **K5**, **K6**, **K7** and **K1** of the register K respectively.

Moreover in the tag-bit controlling circuit **37** said bit B1M="1" read out of the n^{th} decimal denomination of the register M is directly transferred on the output SM via the gate **30** opened by the bistable device **A3** instead of being stepped through the register K.

Therefore it is apparent that the tag bit B1M="1" is recorded in the $(n-1^{st})$ decimal denomination and that the second digit set up on the keyboard is also written in said $(n-1^{st})$ denomination that is the denomination which precedes the denomination where the first digit has been entered.

It is thus clear that the tag bit B1M="1" is shifted from the n^{th} decimal denomination to the $(n-1^{st})$ denomination so as to be relocated any time at the beginning of the last entered digit.

The bistable device **A7** is deenergized by the first timing pulse **T2** occurring after the reading of said tag bit **B1M**. Therefore during the following memory cycles the repetition of the transfer process from the keyboard to the register K for the digit set up on the keyboard is avoided and the first and second digits, included the tag bit B1M="1" which at present is associated with said second digit, circulate in the closed loop formed by the registers K and M.

In a similar manner the following digits of the number are set up on the keyboard and entered into register M. In general, any new entered digit is written in the decimal denomination preceding the denomination of the last entered digit, on account of the fact that the digits are entered beginning from the most significant one and read out of the delay line and processed beginning from the least significant one.

Moreover, any time a new digit is entered via the keyboard, the tag bit B1M="1" is shifted from the last entered digit to said new entered digit to allow the decimal denomination containing the last entered digit to be subsequently recognized.

It is thus apparent that any digit counter is dispensed with in this phase of the computer operation, due to the use of the shiftable tag bits.

It is also apparent that, contrary to the known computers, the operator may set up on the keyboard any number without any care as to its alignment.

For entering the decimal point the operator pushes the key **67** after having entered the units integer digit, whereby a signal V having a duration of a few memory cycles is produced. As the digit indicating signal G1 is absent, the bistable device A7, and thus also the bistable device A3, is not energized, whereby the gate **24** connecting the keyboard to the register K remains closed, and the mechanism for shifting the tag bit B1M="1" to the next following decimal digit is inoperative.

As the bit B1M="1" associated with said units integer digit, which is now the last entered digit, is read out of the memory LDR, a bistable device A80 is energized. The bistable device A80 is thereafter deenergized by the next following clock pulse T1, whereby, assuming this digit has been entered in a certain decimal denomination Cm of the register M, said bistable device will remain energized during the entire digit period Cm. Therefore during the fourth bit period T4 of said digit period Cm a decimal-point indicating bit B4="1" is entered in the stage K8 of the register K via a gate **81**. Said decimal-point indicating bit is thus written in the binary denomination T4 of the decimal denomination occupied by said units digit.

It has been thus explained how a number is entered from the keyboard **65** to the register M of the memory LDR.

In this status **P0**, should the operator set up an address on the keyboard **68** instead of a number on the keyboard **65**, whereby the signal G2 instead of G1 is produced, the four bits H1, H2, H3, H4 representing in this case said address would be transferred via the gate **70** into the stages I1, I2, I3, I4 of the instruction staticisor **16** respectively. Thus the computer receives through the decoder **17** the address Y1 to Y8 of the selected register.

In the manual mode of operation, in the status **P0** the entering of a number and the selection of a register are always followed by the entering of a function via the function keyboard **69**. The actuation of the keyboard **69** generates a signal G3, whereby the four bits H1, H2, H3, H4 which in the present case represent the function set-up on the keyboard, are transferred via a gate **71** into the stages I5, I6, I7, I8 of the staticisor **16** respectively, so as to indicate to the computer, through the decoder **18**, the function F1 to F16 set up on the keyboard. Moreover, whatever said function may be, the beginning of the signal G3 energizes a bistable device A6, whereby in the change-of-status timing circuit **29** the leading edge of the signal A10, produced at the beginning of the next following memory cycle when the generator **44** starts, generates via a gate **83** a timing signal MG which causes the computer to switch to the next following status, said next following status being determined according to the particular instruction at present set up on the keyboard and staticized in the staticisor **16**. The same signal MG deenergizes the bistable device A6, which is therefore effective to prevent the circuit **29** from unduly producing other change-of-status timing signals MG in the following memory cycles occurring during the signal G3. In said next following status, the computer will execute the instruction set up on the keyboard.

Addition and subtraction

The addition and the subtraction of two numbers stored in the registers M and N respectively are accomplished according to the following rules. A true addition is performed when either the signs of the numbers M and N are equal (bistable device A8 is energized) and the instruction at present staticized in F1 (addition) or the signs of the numbers N and M are different (bistable device A8 is deenergized) and the instruction at present staticized is F2 (subtraction). In the other cases a subtraction is effectively performed.

To perform an addition, during a first memory cycle, in which the computer is in the status P5, the two numbers N and M are added together digit by digit, a decimal carry being transmitted to the next higher decimal denomination if the sum digit either is greater than 15 or lies between 10 and 15, the first circumstance being indicated by the presence of a final binary carry R8 produced by summing up the most significant bits B8 and the second circumstance being indicated by the energization of the bistable device **58**. For this purpose the output of the bistable device **58** during the execution of an addition is connected to the summing network **48** via a gate **62**. The result obtained by adding together the two numbers in the above manner is not correct, in that some digits of the result may be greater than nine and therefore have no meaning in the binary-coded decimal code, whereby a radix correction from the binary code to the binary-decimal code is to be performed. To this end during the single memory cycle in which the computer is in the status P5 allotted to the computation of the uncorrected sum a tag bit B1M is recorded in each decimal denomination to indicate the nature of the radix correction to be performed upon the corresponding sum digit, during a following memory cycle (in which the computer is in the status P6) said sum being corrected digit by digit according to the indications given by said tag bits.

More particularly, in the case of the addition, during the second memory cycle, in which the computer is in the status P6, each digit of the sum is corrected from the binary code to the binary-decimal code by adding the filler digit +6 to each digit of the result which in the first memory cycle (while computing the uncorrected sum) had produced a decimal carry.

Therefore the addition is accomplished within two memory cycles, in which the computer is in the status P5 and P6 respectively.

In order to execute the subtraction, during a first memory cycle, in which the computer is in the status P5, the numbers M and N are added together, after having complemented to 15 each decimal digit of the number N. During this cycle a decimal carry is transmitted from a denomination to the next higher denomination only if the sum digit for the first mentioned denomination is greater than 15 (this circumstance is indicated by the presence of a final binary carry R8 from the highest binary denomination T8 of said denomination), no decimal carry being transmitted if said sum digit lies between 10 and 15. For this purpose the gate **62** is held closed for preventing the output of the carry indicating bistable device **58** from being connected to the summing network **48**. The absence of an end decimal carry RF resulting from the addition of the two most significant decimal digits of the numbers M and N respectively indicates in this status P5 that the number M is less than the number N, where as the presence of said final carry RF indicates that the number N is less than the number M.

In the first case, during a following memory cycle (in which the computer is in the status P6) the radix correction is performed by adding either the filler digit +6 or +0 to each digit of the uncorrected sum depending on whether in the status P5 when adding the pair of most significant bits B8 of the corresponding decimal denomination a binary carry R8 had been produced or not. Moreover in the status P6 each digit of the sum, while being corrected, is also complemented to 15 again, whereby the subtract operation is completed within two memory cycles. If, on the contrary, the number N is less than the number M (this circumstance is indicated by the presence of said end carry RF in the status P5) in the status P6 the filler digits to be added to each digit of the uncorrected result are +0 and +10 respectively for the two cases previously considered; moreover in the status P6 the result is not recomplemented, but instead during a new memory cycle (in which the computer is in the status P7) the number +1 is added to the corrected result, thus obtaining a new result which is in turn corrected from the binary to the binary-decimal code during a following memory cycle (in which the computer is in the status P8). Therefore in this case the operation is completed in four memory cycles (corresponding to the four statuses P5, P6, P7 and P8 respectively).

The operation of the computer during the addition and the subtraction will now be described in more detail.

After having aligned the two numbers M and N with respect to their decimal point in the statuses P3 and P14 respectively, and after having examined the signs of the two addends in the status P9, the computer switches to the status P5. During this status the bistable device A8 continues to give an indication as to the agreement of the signs of the two addends as determined in the status P9, whereby in the status P5 the circuit **64** (FIG. 4) produces a signal SOTT if either there is a sign disagreement and the instruction at present staticized is F1 (addition) or there is a sign agreement and the instruction at present staticized is F2 (subtraction), whereas in any other case the circuit **64** produces a signal ADD.

In the status P5 the switching network **36** permanently connects the outputs LN and LM of the registers N and M to the two inputs **1** and **2** of the adder **72** respectively, the output **3** of the adder to the input **13** of the register K and the output **14** of the register K to the input SN of the register N. Moreover the output of all the memory registers, except the register N, is corrected to the respective input. Therefore in this status, which lasts a single memory cycle, the contents of the register M, without being destroyed, is added to the contents of the register N, the latter contents having been either complemented to **15** digit by digit via the complementer **54** or not depending on whether the signal SOTT or ADD is present, the result being written in the register N via gate **55**, while the contents of all the other registers is regenerated so as to remain unchanged.

More exactly, the connection between the inputs **1** and **2** of the adder and the outputs LM and LN of the registers M and N exists only during the bit periods T5, T6, T7 and T8 of each digit period.

During the remaining bit periods T1, T2, T3 and T4 the switching network **36** directly connects the output of the register N to the input of the register K, so as to bypass the adder **72**, whereby the bits B1, B2, B3, B4 of each decimal denomination, which are tag bits to be held unmodified in this phase, are regenerated.

On the contrary during the bit periods T5, T6, T7, T8 of the generic n^{th} decimal denomination the bits B5, B6, B7, B8 respectively of the corresponding decimal digit of the number M are added to the bits B5, B6, B7, B8 respectively of the corresponding decimal digit of the number N (the four last mentioned bits being inverted by the inverter **53** if the signal SOTT is present), each pair of corresponding bits being fed to the adder along with the binary carry produced by adding the next preceding pair of bits and staticized in the bistable device A5, whereby the added **72** produces in each digit period during the bit periods T5, T6, T7 and T8 respectively, four bits representing a decimal digit of the uncorrected sum. Due to the previous explained connection of the register, said uncorrected sum digit, assuming it has been produced by adding two addend digits stored in the n^{th} decimal denomination of the registers M and N respectively, is recorded in the $(n-1^{st})$ decimal denomination of the register N.

During said generic n^{th} digit period, and more exactly at the end of the last bit period T8 thereof, the binary-carry staticizing bistable device A5 is as usually energized or not depending on whether the sum of the last pair of bits B8 has generated a final binary carry R8 or not. The bistable device A5 thereafter remains as usually in the energized state until it receives from the bistable device A4 the new binary carry produced by summing up the next following pair of bits, which in this case are the first bits B5 of the next following digit period $C(n+1)$. Therefore it is apparent that the bistable device A5 is adapted to feed said final binary carry R8 of the n^{th} decimal denomination to the adder **72** when the adder receives the first pair of bits B5 of the $(n+1^{st})$ decimal denomination. As said final binary carry indicates also the presence of a decimal carry, it is clear that said bistable device A5 is also adapted to transmit the decimal carry between said two decimal denominations. This happens both in the case of addition (signal ADD is present) and in the case of subtraction (signal SOTT is present). Moreover in the case of addition, but not in the case of subtraction, gate **62** is opened during the bit period T1 immediately following said bit period T8 for connecting the bistable device **58** to the bistable device A5, whereby in the case of addition when the adder receives the first pair of bits B5 of the $(n+1^{st})$ decimal denomination the bistable device A5 feeds a decimal carry to the adder not only if the sum digit in the n^{th} denomination was greater than fifteen but also if said sum digit was between ten and fifteen.

Therefore, in every case, in the status P5 the fact that the bistable device A5 is energized during the bit period T1 of the $(n+1^{st})$ digit period indicated that a carry has been transmited from the n^{th} to the $(n+1^{st})$ decimal denomination. In said bit period T1 the tag bit controlling circuit **37** causes a tag bit B1M="1" to be written into the $(n+1^{st})$ decimal denomination of the register M via a gate **85** if said decimal carry has been produced in the n^{th} decimal denomination. The same happens for each one of the successive digits to be added. It is to be noted that said tag bit is effectively written via gate **85** in the proper denomination because writing in the register N is now effectively delayed one digit period with respect to writing in the register M due to the fact that in the present status the contents of the register N recirculates through the register N and the shift register K while the contents of the register M recirculates only through the register M itself.

Furthermore, it is to be noted that, due to the aforesaid connection of the registers N, K and M (register M has its input directly connected to its output, while register N has its input and its output connected to the output and to the input respectively of the register K, which is long one digit period) at the end of the status P5, which lasts a single memory cycle, the uncorrected result of the addition, stored in the register N, will appear as delayed one digit period with respect to the contents of the register N.

Only in the case of subtraction (signal SOTT is present) in the first bit period T1 following the digit period in which the last (most significant) pair of decimal digits of the numbers M and N has been added, the decimal carry signal, if any, produced by adding said last pair of decimal digits is sent via gate **63** to energize the bistable device RF. The bistable device RF will thereafter indicate during the following memory cycles the existence of said end carry, whereby the circumstance that said bistable device RF is either energized or not will indicate whether the number N was less than the number M or not.

It is to be noted that gate **63** may be opened only after disappearance of the signals **A1** and **A0** indicating the length and position of the number N and M, whereby the bistable device is responsive only to the end carry produced by adding the last pair of digits.

Upon completion of this summation cycle, the leading edge of the signal **A01** produces via gate **87** in the circuit **29** a change-of-status timing pulse MG which causes the computer to switch to the next following status. This status, as determined by the logic network **27**, is the status **P6**, which lasts a single memory cycle and is spent for the correction of the sum.

The status **P5** is always followed by the status **P6**, whatever the internal conditions of the computer may be.

In the status **P6** the switching network **36** connects the register M and the register K so as to build up a closed loop, whereby the contents of the register M is delayed one decimal denomination with respect to the register N. Since in the preceding status **P5** the contents of the register N had been delayed the same amount with respect to the register M, the two numbers M and N are thus restored into their previous alignment with respect to the decimal point. Moreover the switching network **36** connects the inputs **1** and **2** of the adder to the output LN of the register N and to the output **32** of a filler digit generator **31**, and the output **3** of the adder to the input SN of the register N. As previously explained, due to the relative displacement of the numbers stored in the registers M and N, in this status **P6**, when beginning to read out of the delay line the n^{th} decimal denomination of the register N, the tag bit **B1M** is read out of the delay line, this tag bit indicating what kind of radix correction is to be performed upon said n^{th} digit of the uncorrected sum stored in the register N. More particularly the reading signal **LB1M** produced by reading said tag bit from the memory LDR either energizes the bistable device **A7** or not depending on whether its value is "1" or "0," said bistable device **A7** being thereafter deenergized at the beginning of the next following clock pulse **T1**, whereby during the entire n^{th} digit period the bistable device **A7** indicates what kind of correction is to be performed upon the uncorrected sum digit stored in said n^{th} denomination of the register N.

More particularly, if an addition is being performed (signal ADD is present), the bistable device RF is surely deenergized, because, as previously stated, the existance of an end carry RF produced during the status **P5** by adding together the most significant pair of digits has no relevance in the case of addition.

In the case of addition, in the status **P6** the output S of the addition network **48** is connected to the output **3** of the adder **72** via gate **55**, whereby the corrected sum produced in said status **P6** is not recomplemented. Moreover, while feeding the input **49** of the addition network **48** with the digit of the n^{th} decimal denomination of the register N (uncorrected sum) via gate **52**, the filler digit generator **31** simultaneously feeds the input **2** with the filler digit **6**, whose code representation **B5**=0, **B6**=1, **B7**=1, **B8**=0 is produced via gate **33** provided the bistable device **A7** is simultaneously in the energized state; if on the contrary the bistable device **A7** is deenergized, generator **31** feeds the input **2** with the decimal digit 0, which is represented by four binary zeroes.

In the case of subtraction (signal SOTT is present) and if in the preceding status **P5** no end decimal carry RF has been produced, whereby the bistable device RF also in this case is deenergized, in the status **P6** the output S of the addition network **48** is connected to the output **3** of the adder **72** via gate **56** and inverter **57**, whereby each bit **B5**, **B6**, **B7**, **B8** of the corrected sum is inverted (and so the decimal digit represented by said four bits is recomplemented to 15) before being rewritten into the register N. The radix correction of the sum is accomplished by adding to each digit of the uncorrected sum either the filler digit **6** via gate **33** of the filler digit generator **31** or 0 as in the previous case.

If, on the contrary, in the case of subtraction, the signal RF is present to indicate that in the preceding status **P5** an end decimal carry had been produced, the corrected sum produced by the adder **72** in the status **P6** is written into the register N via gate **55** without complementing. Moreover in this case while feeding the addition network **48** via gate **52** with the bits **B5**, **B6**, **B7**, **B8** of the uncorrected sum digit contained in the generic n^{th} digit period of the register N, the filler digit generator **31** simultaneously produces via gate **34** the bits **B5**=0, **B6**=1, **B7**=0, **B8**=1 representing the decimal number 10 if the bistable device **A7** is in the deenergized state during said digit period; if on the contrary the bistable device **A7** is energized, the decimal digit 0, represented by four binary zeroes, is fed.

In all the three aforesaid cases (addition, subtraction with M less than N, subtraction with N less than M), during the status **P6** the leading edge of the signal **A01** produces, via the gate **87** of the circuit **29**, a change-of-status timing pulse MG which causes the computer to switch to the next following status.

So in the first two cases the addition, respectively the subtraction, is completed, whereby the logic network **27** designates as the next following status either the status **P17** (extract the next following instruction) if the computer is preset for the automatic mode of operation and the instruction **F1** (addition) of **F2** (subtraction) is at present staticized, or the status **P18** (begin to print out the first addend) if the computer is preset for the manual mode of operation and the instruction **F1** (addition) or **F2** (subtraction) is at present staticized.

On the contrary, in the third case, in which the bistable device RF remains energized, the status **P6** is followed by the status **P7**, in which the number +1 is added to the result stored in the register N and by a status **P8** in which the digits of the new result thus obtained are corrected from the binary code to the binary decimal-code, the operation of the computer in said statuses **P7** and **P8** being similar to the operation in the statuses **P5** and **P6** respectively. In the status **P8** the leading edge of the signal **A01** indicating that there are no more digits to be added, causes the computer to switch (see FIG. 7) to the next following status, which is either the status **P17** or the status **P18** or another status as previously explained.

As to the sign of the result, in the status **P6** the sign bits recorded in the register N are regenerated without modification if in the status **P5** no end decimal carry RF has been produced, whereas they are inverted by obvious means not shown in the drawings before being rewritten into the delay line LDR if the final carry RF is present.

According to a second embodiment of the computer according to the invention, not shown in the drawings, the addition an the subtraction are performed according to the following rules.

In a first memory cycle (in which the machine is in the status **P40**) the number M is added to the number N after having complemented each digit of the number N to 15, for the only purpose of determining, on the basis of the existance of an end decimal carry RF, whether N is greater than M or not.

The operation of the computer in this status **P40** is quite similar to the operation in the status **P5** according to the first embodiment when the signal SOTT was present, apart that now the register N is not connected to the register K but has its output connected to its input via the adder **72**.

During a second memory cycle (in which the computer is in the status **P50**) the number M is added to the number N, the several digits of the greater one of the two numbers M and N being either complemented to 15 or not depending on whether a subtraction or an addition is being performed. For this purpose the switching network **36** connects either the output LN of the register N and the output LM of the register M to the inputs **1** and **2** respectively of the adder **72** or vice versa depending on whether said signal RF is present or not, the input **1** being anyway connected to the input **49** via the complementer **54**. In a third memory cycle (in which the computer is in the status **P60**) the correction from the binary code to the binary-decimal code is performed by adding the filler digit +6 to each uncorrected sum digit which has produced a final binary carry **R8** and the filler digit +0 to each other uncorrected sum digit. Moreover the digits of the result are recomplemented to 15 if a subtraction is being performed.

The modifications to be made in the adder shown in FIG. 4 to make it capable of operating according the preceding rules are obvious to those skilled in the art.

From the foregoing it is apparent that whenever the instruction staticisor **16** staticizes the instruction Y, **F1** (addition) or Y, **F2** (subtraction), the computer is adapted under the control of the sequencing circuit **26** to automatically go through a sequence of statuses which, according to the second embodiment of the adding device of the computer, is as schematically shown in FIG. 8*a*.

More particularly, starting either from the status **P0** in which said instruction is set up on the keyboard in the manual operation or from the status **P17** in which said instruction is extracted from the memory LDR in the automatic operation, the addition (or subtraction) sequence comprises:

status **P2**, wherein the contents of the register Y addressed by said instruction is transferred into the register M;
statuses **P3** and **P14**, wherein the numbers stored in the registers M and N respectively are aligned so as to have their decimal point located in the first decimal denomination **C1**;
status **P9**, wherein the two numbers M and N are examined to determine whether their algebric signs are in agreement;
status **P40**, wherein the two numbers M and N are examined to determine whether number M is greater than number N or not;
status **P50**, wherein the two numbers M and N are added together;
status **P60**, wherein the radix correction for the sum so obtained is performed.

After this sequence, the computer, if preset for the automatic mode of operation, automatically reverts to the status **P17**, wherein the next following instruction is extracted; if preset, on the contrary, for the manual mode of operation, it goes through the sequence of statuses **P18**, **P19**, **P22** during which the number Y is printed out and thereafter it reverts to the status **P0** wherein the next following instruction is set up on the keyboard.

MULTIPLICATION AND DIVISION

If the instruction at present staticized in the staticisor **16** is Y, F3 (multiplication) the sequence of statuses the computer goes through, starting either from the status **P0** (if in manual operation) or from the status **P17** (if in manual operation) or from the status P17 (if in automation operation) is as follows (FIG. 8*b*):

status **P2** (lasting one memory cycle) wherein the number stored in the register Y (multiplicand) addressed by said instruction is transferred into the register M;
status **P3**, wherein the number stored in the register M (multiplicand) is repeatedly shifted until its first (least significant) integer digit containing the decimal point bit **B4**="1", reaches the first decimal denomination **C1** of the register M;
status **P14**, wherein the number stored in the register N (multiplier) is repeatedly shifted (one digit period for each memory cycle) until its most significant digit reaches the first decimal denomination C1 of the register N;
status **P9** (lasting one memory cycle) wherein the two numbers to be multiplied are examined as to sign agreement, while the contents of the register N (multiplier) is transferred into the register R for allowing the register N to subsequently accumulate the product;
status **P40** (lasting one memory cycle) wherein the two operands are examined to determine which is the greatest one (this has no relevance when multiplying, but rather only when dividing);
status **P10** (lasting one memory cycle) wherein the digit of the multiplier which is stored in the decimal denomination occupied by the decimal point of the multiplicand is diminished one unit, while the multiplier itself is delayed (that is shifted toward the most significant denomination one digit period;
status **P50** (lasting one memory cycle), wherein the multiplicand M is added to the number stored in the accumulator N;
status **P60** (lasting one memory cycle), wherein the radix correction of the sum obtained in the preceding status is performed.

From this status **P60** the machine reverts into the status **P40** for repeating the partial sequence **P40**, **P10**, **P50**, **P60**, which partial sequence is repeated n times if n is the most significant decimal digit of the multiplier. It is to be noted that the numbers stored in the registers R, N and M are delayed one digit period, that is shifted one decimal denomination toward the most significant denomination, in the statuses, **P10**, **P50**, and **P60** respectively whereby after each one of said partial sequences **P40**, **P10**, **P50**, **P60** said three numbers are restored into their previous alignment. After the n^{th} of said partial sequences, in order to shift the multiplier (register R) and the partial product (register N) one decimal denomination toward the most significant denominations, a reduced partial sequence comprising the statuses **P40**, **P10**, **P50** is executed. In the status **P50** of this reduced partial sequence, contrary to the normal operation of the computer in the status **P50**, the switching network **36** does not connect the register M to the adder **72**, whereby the number N is shifted without being altered.

Thereafter m partial sequences **P40**, **P10**, **P50**, **P60** are executed as previously explained, if m is the second most significant digit of the multiplier, and so on.

By examining in more details the operation of the computer, it is to be noted that in the status **P9** the multiplier is transferred from the register N to the register R via a binary inverter, whereby each decimal digit of the multiplier itself is complemented to 15.

In the status **P10** the switching network **36** connects the output LR of the register R to the input **1** of the adder **72**, whose output is connected to the input **13** of the register K, whose output **14** in turn is connected to the input SR of the register R so as to build up a closed loop. As the second input **2** of the adder **72** receives no signal, the contents of the register R recirculates in said loop without being altered and is therefore delayed one digit period in each memory cycle. Moreover, under these conditions said loop is adapted to act as a counter in the way previously explained in the general description, in order to count the adding cycles performed for each digit of the multiplier. More particularly it will be remembered that for having said loop to act as a counter, it is necessary to feed the binary-carry storing bistable device **A5** with a counting pulse (that is, to simulate a binary carry) in the bit period in which the minimum-weight bit contained in the counter is fed into the adder. In the present case this bit will be the bit **B5** of that decimal digit of the multiplier which is now to be modified by means of the counting pulses. In the present case, when reading the decimal point bit **B4**="1" of the regitser M, the bistable device **A5** is energized to simulate said binary carry, which carry will be fed to the adder **72** concurrently with the first bit **B5** of that digit of the multiplier which, having been complemented to 15, is now processed. Therefore the last mentioned digit will be increased one unit during each partial sequence of statuses **P40**, **P10**, **P50**, **P60** as well as

during each reduced partial sequence of statuses **P40, P10, P50.**

Therefore, if *n* is the digit of the multiplier now considered, after *n* partial sequence **P40, P10, P50, P60** said digit of the multiplier will become 15. In the meantime, the computer begins to repeat once more said partial sequence, whereby in the status **P10** said digit of the multiplier becomes 16, thus producing a final binary carry **R8** coming out from the last bit period **T8** of said digit of the multiplier. This carry energizes the bistable device **A6**, which during the following status **P50** will affect both the switching network **36** for preventing the register M from being connected to the adder and the logic circuit **27** for causing said status **P50** to be followed by status **P40** instead of status **P60**, whereby the partial sequence of statuses the computer goes through in this case will be the reduced sequence **P40, P10, P50** in which the partial product produced in the register N is not altered and the partial product itself along with the multiplier are shifted. Immediately after said binary carry **R8** has been produced, the bistable device **A5** will be deenergized by the clock pulse **T2** so as to clear out said carry stored therein, for preventing said carry from being unduly transmitted to the other denominations of the multiplier, because said other denominations must not be modified in this phase of the multiplication.

It is to be noted that, due to the shifting of the multiplier R during said reduced partial sequence **P40, P10, P50**, the digit of the multiplier next following the digit just considered is shifted into the denomination corresponding to that denomination of the register M which contains the decimal point of the multiplicand and that said relative alignment of the multiplier with respect to the multiplicand will remain unchanged throughout the following partial sequences **P40, P10, P50, P60** until also the partial product of said next following digit and the multiplicand will be computed and accumulated, whereby the decimal point bit **B4**="1" of the multiplicand M acts as a mark for identifying the digit of the multiplier R which is now to be considered.

From the foregoing it is further apparent that the reduced partial sequence **P40, P10, P50** executed after completion of the computation of the partial product relating to the last (least significant) digit of the multiplier R will cause said last digit to be shifted one denomination beyond the decimal point of the multiplicand M. Therefore, in the following status **P40**, during the digit period wherein the decimal point bit **B4** of the register M is read out of the memory LDR, no digit-indicating bit **B2**="1" will be concurrently read out in the register R. Upon occurence of this circumstance the bistable device **A9** will be energized by the reading signal produced by reading out said decimal point bit, whereby the bistable device **A9** will affect the logic circuit **27** so as to prevent it from determining as the next following status the status **P10**. Thus the multiplying operation ends. The next following status will be either the status **P17** (extract the next instruction) if the computer is preset for automatic operation or the status **P18** (first status of a sequence **P18, P19, P22** wherein the multiplicand Y is printed out) if the computer is preset for manual operation.

In a similar way the division is performed according to the repeated subtraction method.

Entering a program through the keyboard

Having preset the commutator **23** so as to produce signal IP ("entering program") the operator sets up on the address keyboard **68** and on the function keyboard **69** the successive instructions of the program to be entered.

Since entering a program via the keyboard into the program registers I and J is similar to entering data via the keyboard into the register M, which operation has been previously described, no further description is deemed to be necessary to those skilled in the art.

After having entered the program into the memory, by actuating a push button AUT the operator may start the automatic execution of said program.

Extracting an instruction

The program having been entered in the memory LDR, actuation of a push button AUT starts the program execution.

The actuation of said button AUT sets the computer in the status **P17**, in which the switching network **36**, beside connecting the input of each memory register to the respective output so as to continuously regenerate its contents, connects the output of the register I or J (or any other instruction register involved in the transfer operation) to the instruction staticisor **16** only during the digit period in which the instruction to be extracted and executed is being read out of the delay line, said digit period being identified by the energization of the bistable device **A3**.

More particularly, in the first memory cycle occurring during the actuation of said push button AUT, the synchronizing bit **B1R**="1" which starts the oscillator **45** at the beginning of the first bit period **T1** of the first digit period **C1** energizes the bistable device **A3**, which thereafter is deenergized at the end of said bit period **T1**. Moreover the beginning of the signal AUT energizes the bistable device AI, which when energized, causes the instruction register I to be addressed and selected via the switching network **36**, the instruction register J being in turn addressed and selected when said bistable device AI is deenergized. The bistable device AI acts as an address counter to sequentially address the successive instruction registers I, J, since the program is normally executed by first sequentially executing all the successive instructions stored in the register I, then all the successive instructions stored in the register J.

Therefore during said first digit period C1 the output line **LI** of the instruction register I is connected to the instruction staticisor **16**, whereby the eight bits **B1** to **B8** of the first instruction are written in the eight stages **I1** to **I8** respectively of the staticisor **16**, wherein they are staticized until, after execution of said first instruction, the next following one is extracted.

Moreover in said first digit period **C1**, as the bistable device **A3** is energized, the clock pluse **T8** energizes the bistable device **A9**, which is thereafter deenergized by the next following clock pulse **T8**. Therefore the bistable device **A9** is adapted to identify, by being in its energized state, the digit period next following the digit period of the instruction being now extracted.

As said bistable device **A9** is energized, the tag-bit controlling circuit **37** causes a tag bit **B1N**="1" to be written via gate **91** into the second decimal denomination **C2** of the register N, said tag bit **B1N** being a mark which will be used to identify said next following instruction to be extracted, which in this case is the second instruction.

Moreover, as said bistable device **A9** is energized, the clock pulse **T1** of sad second digit period **C2** energizes the bistable device **A6** to indicate that the instruction to be extracted has been recognized and extracted. Therefore, at the end of the memory cycle, the leading edge of the signal **A10** causes the gate **83** of the circuit **29** (FIG. 7) to produce a change-of-status timing signal MG which causes the computer to switch to the next following status, this status being identified by the logic network **27** on the basis of the instruction just extracted and staticized. This next following status is the first status of a sequence of statuses during which said instruction is executed.

At the end of the execution of said first instruction, the computer is caused by the sequence control circuit **26** to automatically revert to the status **P17**, wherein the second instruction is extracted, and so on.

In general, at the end of the sequence of statuses in which the n^{th} instruction has been executed, the computer automatically reverts to the status **P17**, under the control of signals indicating the completion of the corresponding operation. In the status **P17**, which lasts a single memory cycle, the delay line is scanned for searching in the register I or J the instruction to be extracted, which is the $(n+1^{st})$ instruction. Recogni.ion of this instruction is made on the basis of the presence of the tag bit **B1N**="1" in the $(n+1^{st})$ decimal denomination of the register N. Upon reading out of the delay line said tag bit **B1N**, the bistable device **A3** is energized to identify the digit period in which the instruction to be extracted is delivered at the output of the delay line LDR. Under the control of said bistable device **A3**, the switching network **36** connects the output of the register I or J to the instruction staticisor **16** only during this digit period. Due to energization of the bistable device **A3**, the bistable device **A9** is subsequently energized to identify the next following digit period $C(n+2)$, whereby in the tag-bit controlling circuit **37** a tag bit **B1N**="1" is written via gate **91** into said digit period $C(n+2)$, whereby said tag bit is shifted from the $(n+1^{st})$ instruction being at present extracted to the next following $(n+2^{nd})$ instruction to be extracted.

Should the aforesaid n^{th} instruction be the last (22^{nd}) instruction of the register I, the bistable device **A9**, which in any case in the status **P17** is always energized during the only digit period next following the digit period of the instruction being at present extracted, will happen to be energized during the first digit period **C1**, in which the synchronizing bit **B1R**="1" starting the next following memory cycle is read out of the memory. The concurrence of said two events (energization of bistable device **A9**, reading out the start bit **B1R**) causes the instruction-register addressing bistable device **AI** to switch so as to be deenergized, whereby in the following statuses **P17** the instruction register J instead of I will be addressed and selected, The tag-bit controlling circuit **37** causes as usually a tag bit **B1N**="1" to be written via gate **91** into the decimal denomination (**C1** in the present case) next following the instruction being at present extracted, whereby the first instruction of the register J will be thereafter extracted.

It is thus apparent that the use of a tag bit shiftable along the delay line allows the register I and J to be sequentially scanned for extracting one at a time the successive instructions of the program stored therein, the same tag bit being effective upon reaching the end of an instruction register to advance an instruction-register selecting counter AI for addressing the next following instruction register.

Jump

According to an embodiment of the invention, in the jump instruction the four bits **B5, B6, B7, B8**, which are used, as in any other instruction, to represent the function part **F12**, of the instruction itself, are

$$B5=B6=B7=B8="1"$$

The presence of this four-bit combination in an instruction of the program indicates that the instruction itself is concerned with a jump operation during the execution of the program. In this instruction, the bits **B1** and **B2** represent an address, while the bits **B3** and **B4** are used to further specify the nature of the instruction.

More particularly, if **B3=B4**="1," the instruction is not a true instruction, because, upon being entered into the staticisor **16** the instruction does not cause the computer to perform any operation. On the contrary, this instruction is merely a "reference instruction" used as a reference point within the sequence of program instructions, whereby among the 44 instructions of the program stored in the registers I and J it is possible to establish some reference points, each one represented by a reference instruction. There are four different types of reference instructions depending on the value of the bits **B1** and **B2** of the reference instruction, which bits define the "address" of this reference instruction. Each reference instruction marks the beginning of a subroutine, whereby the reference instructions have the function of marks dividing the program into subroutines.

If **B3**="0," the instruction is a true jump instruction, the jump being conditional or unconditional depending on whether **B4** is equal to "1" or "0."

Each one of said jump instruction, having been extracted from the delay line and staticized in the staticisor **16** during the status **P17** of the computer, as any other instruction, causes the computer to switch to the status **P23**, in which the program registers I, J are scanned to search a reference instruction having the address specified in said staticized jump instruction, that is, having the bits **B1** and **B2** equal to the corresponding bits of said jump instruction. More particularly, in this status **P23** during a first memory cycle the successive instructions stored in the first instruction register I are read out of the delay line and, besides being regenerated, are fed to a comparator, not shown in the drawings and well known in the art. This comparator is adapted to receive each series of eight bits representing an instruction, and to produce an output signal if said instruction is found to be equal to the required reference instruction, that is to have all the bits **B3, B4, B5, B6, B7** and **B8** equal to "1" and the bits **B1** and **B2** equal to the bits **B1** and **B2** of the jump instruction at present staticized.

For instance said comparator may be made of a binary comparator having one input connected to the output of the instruction register at present addressed and selected for receiving said series of eight bits of each scanned instruction and the other input fed by a logic network mechanizing the function

$$T1\cdot I1+T2\cdot I2+T3+T4+T5+T6+T7+T8$$

wherein **T1** to **T8** are the clock pulses produced by the generator **44** and **I1** and **I2** are the outputs of the two corresponding stages of the instruction staticisor **16**, said comparator being adapted to produce an output signal upon receiving at its inputs a pair of simultaneous bits having different values. Said output signal is used to deenergize a bistable device which is energized by the clock pulses at the beginning of each digit period. It is thus apparent that at the end of each digit period this bistable device is energized or not depending on whether the instruction at present scanned coincides with the required reference instruction or not.

If coincidence occurs, said bistable device causes the tab-bit controlling circuit to write a tag bit **B1N**="1" in the next following decimal denomination to indicate that the next instruction to be extracted (first instruction of the required subroutine) is the instruction stored in said denomination. For the purpose of extracting and staticizing said first instruction of the subroutine, upon detecting said coincidence the computer switches into the instruction-extract status **P17**, whereby execution of said subroutine begins.

In order to revert to the interrupted main program after completion of said subroutine, it is possible either to put a suitable jump instruction at the end of said subroutine according to a known technique, or to use a tag bit **B1U**="1" which is recorded in the register U when interrupting said main program, so as to mark the instruction of the register I or J last executed in said main program. For this purpose in the status **P17**, if a jump instruction is being extracted, contrary to the procedure previously explained, the tab bit **B1N**="1" is not shifted to the next following decimal denomination of the register N, but instead is transferred into the corresponding denomination of the register U by means obvious to those skilled in the art and not shown in the drawings.

According to a feature of the invention, the reference instructions may be used also in the manual mode of operation for executing certain subroutines. For this pur-

pose the keyboard **22** is provided with four subroutine keys **V1**, **V2**, **V3**, **V4**, corresponding to the four possible "addresses" of the reference instructions respectively, whereby each subroutine key **V1** to **V4** has associated therewith an "address" represented by the two bits **B1** and **B2**.

In the manual operation, while the computer is in the status **P0** waiting for the setting up of a new datum and a new instruction on the keyboard **22**, the operator may push one of said four subroutine keys **V1** to **V4**. Actuation of one of said four keys causes the bits **B3**=**B4**="0" and **B5**=**B6**=**B7**=**B8**="1" to be written into the binary stages **I3** to **I8** respectively of the instruction staticisor **16** via a circuit not shown in the drawings, and the address bits **B1** and **B2** corresponding to said key to be written into the stages **I1** and **I2** respectively. Therefore it is apparent that in the status **P0** the actuation of one of the subroutine keys **V1** to **V4** simulates the extraction of an unconditional-jump instruction from the delay line to the instruction staticisor **16**. Moreover actuation of said subroutine key causes the computer to switch to status **P23**, in that a signal issued from the keyboard directly sets the bistable devices **P1** to **Pn** into said status by energizing the bistable device **P23** and deenergizing the other bistable devices. As previously explained, in this status **P23** the program registers I and J are searched for a reference instruction having the same address **B1**, **B2** of the subroutine key now actuated, and upon finding said reference instruction the computer automatically switches to the status **P17** for extracting the first instruction of the subroutine headed by said reference instruction.

As the execution of this subroutine must be automatic, the actuation of the subroutine keys **V1**, **V2**, **V3**, **V4** must cause the commutator **23** to switch from the position PM (manual) to the position PA (automatic). Therefore it is apparent that said commutator may be conveniently replaced by a bistable device which is energized upon depressing said subroutine key and deenergized upon terminating said subroutine.

Splitting the memory registers

According to an embodiment of the invention, the registers Q, U, Z, D, E may be split into two parts for storing two short numbers. For this purpose a tag bit **B1Z**="1" is recorded as a permanent mark in the first binary denomination (bit period) of a fixed decimal denomination (digit period) of the register Z, for instance the denomination **C12**.

A bistable device, not shown in the drawing, is energized upon reading the synchronizing bit **B1R**="1" which starts the oscillator **45** at the beginning of each memory cycle and is there after deenergized upon reading said fixed tag bit **B1Z**="1," whereby the bistable device is adapted to identify the first part of each memory cycle and distinguish it over the second part thereof, that is to identify the first part of each memory register and distinguish it over the second part thereof.

Since each instruction contains four address bits **B1** to **B4**, it is possible to use the three bits **B2** to **B4** for identifying one out of the eight addresses **Y1** to **Y8** of the eight addressable registers Q, U, Z, D, E, M, N, R, the remaining bit **B1** being used for addressing either the first or the second part of the register concurrently addressed by said three bits **B2** to **B4**.

The splittable registers Q, U, Z, D, E are never directly involved in arithmetic operations. Otherwise stated, their contents (apart from the tag bits **B1**) is never directly modified, in each memory cycle these contents being either regenerated without modification or transferred to or from the registers M or N.

Therefore each one of the two parts of each register Q, U, Z, D, E may be addressed and selected by the switching network **36** under the control of the address bit **B1** at present staticized in the instruction staticisor **16**. More particularly, if said staticized bit is **B1**="1," the switching network **36** connects the splittable register Q, U, Z, D or E at present addressed by the staticized instruction to either the register N or the register M (depending on the function part of said staticized instruction) only when said bistable device is energized, whereby the transfer operation is performed only to or from the first part of said splittable register, whereas if said staticized bit is **B1**="0" the connection is effected only when said bistable device is deenergized, whereby the transfer operation is performed only to or from the second part of the splittable register.

It is obvious that any transfer operation to and from a selected part of a splittable register must be preceded by suitable aligning operations performed on the number stored therein.

In the embodiment considered in the general description, each address key was effective, when actuated, to enter four address bits **B1** to **B4** into the computer. In another embodiment each address key is effective to enter only the three address bits **B2** to **B4** used to address a register, a separate split key being provided for entering the remaining address bit **B1**, whereby any part of any splittable register may be normally addressed through the keyboard.

In an alternative embodiment, the address bit **B1** may be effective, depending on its value, to cause the transfer operation to begin upon reading either the start bit **B1R** (beginning of the memory cycle) or the tag bit **B1Z** (beginning of the second half of the memory cycle), in both cases the transfer operation being continued until the end of the cycle.

In another preferred embodiment of the invention, the memory cycle lasts 24 digit periods instead of 22 as heretofore described, each register being able to store either a 22-digit number or two 11-digit numbers. In this case the digit periods **C12** and **C24** are void, in order to give the computer sufficient time to detect overflow during the arithmetic operations. This arrangement entails modifications which are obvious to those skilled in the art. It is to be pointed out that the lengthening of the memory cycle to 24 digit periods involves only a modification of the number written into the register K at the beginning of the start-computer status **P21**, because, due to the use of the tag bits in the delay line, no digit counter is used in the normal operation of the computer.

Program card

According to an embodiment of the invention, the computer is provided with a device for recording and reading data and instructions on record cards, for instance magnetic cards.

It has been previously explained how the data and the program instructions may be set up on the keyboard and stored into the delay line registers.

Having been so stored in the computer via the keyboard, the data and the program are made available for controlling the computer.

Moreover said data and instructions having been set up on the keyboard may be read out of the delay line and recorded on a card for subsequent use, whereby the operator may prepare a card file for subsequent use.

According to a feature of the invention, each card has sufficient capacity to store at least an entire program. Otherwise stated, it has a capacity not less than the program registers of the computer.

In a preferred embodiment, the card may store the contents of the five memory registers I, J, Z, D, E. The registers I and J are permanently allotted to store program instructions. Each one of the registers Z, D, E, being splittable, may contain either a 22-digit number or two 11-digit numbers or 24 program instructions or a 11-digit number and 12 program instructions, whereby according to the present embodiment of the invention also the registers Z, D, E, may be used, either partially or totally, as program registers.

The storage capacity of a card being related to the storage capacity of the program registers in the above manner, it is apparent that by merely reading a card into the computer the operator may have immediately available any desired program, the only operation required being inserting the card into the reading unit. This gives substantial advantages especially in the manual mode of operation. As a matter of fact, since the operator in the manual operation may institute by means of the subroutine keys **V1**, **V2**, **V3**, **V4** the automatic execution of any subroutine, by merely introducing a properly coded card and then pushing a subroutine key the computer may be caused to perform any desired operation, whereby the computer may be regarded as being provided with an unlimited number of function keys.

Otherwise stated, the computer comprises, besides the function keys of the keyboard **69**, four function keys **V1** to **V4** whose function may be changed by associating therewith a different program card.

More particularly each subroutine key has a fixed four-bit code combination associated thereto and corresponding to a certain setting of the code bars in the keyboard decoder. Actuation of said key causes the computer to search the program registers for a reference instruction having the same code of said key. Upon finding said reference instruction, which marks the beginning of a program subroutine, the computer begins to execute said program subroutine. If said code combination is used to identify in the program stored on a first card a subroutine controlling the computation of the sine and in the program stored on a second card a subroutine controlling the computation of the cosine, for instance, then said key will be given the meaning of "sine key" and "cosine key" upon reading into the computer said first card and said second card respectively.

Therefore by first manually entering for instance said first card into the computer and then depressing said subroutine key, the sine of a datum either previously set up on the keyboard or previously entered on the memory LDR and now addressed via the keyboard is computed.

Each card **150** is made (FIGS. 9 and 10) of a flexible sheet having on at least one face a strip of magnetizable material which constitutes a recording track, the opposite face being adapted to bear visible designations pertaining to the information recorded in coded form on said recording track.

The path for the card is defined by a pair of guides **144**, **115** between an inlet opening **113** and outlet opening **144** of the computer frame.

Two driving rollers **116**, **117** are located along said path for cooperating with pressure rollers **118**, **119** respectively in order to feed the card along said path.

The driving rollers **116**, **117** are connected, by gear means not shown in the drawings, to a motor **120**, which is also adapted to drive the movable parts of both the printing unit **103** and the keyboard decoder **101**.

The pressure roller **119** is pivoted on oscillating arms **121** journalled to an axis **122** and spring urged against the roller **117**.

Also pivoted around said axis **122** is an eccentric hub **124** on which an oscillating arm **123** is mounted. Arm **123** bears a read-write magnetic head **129** spring urged against roller **117**.

By rotating the eccentric hub **124** through an adjusting screw the position of the head along the path of the card may be adjusted.

Arms **125**, **126** also pivoted around axis **122** have journalled thereto a first sensing roller **126** lying before the magnetic head along the card path and a second sensing roller **128** respectively, lying beyond the magnetic head.

The sensing rollers **126** and **128** are urged by springs **130**, **132**, toward the path of the card, whereby when card is absent they partially enter two corresponding openings of the guides **114** and **115** so as to lie in said path, to an extent limited by an abutment **131** which engages the end of an adjusting screw **133**, **134** born by said arms respectively.

The card **150**, upon passing under the sensing rollers **126**, **128** causes them to be raised, so as to counter-clockwise rotate the arm **125**, **127** respectively.

An arm **135** also pivoted around axis **122** is provided with a first projection **136** adapted to engage the actuating button of an electrical switch **137** against which it is urged by a spring **139** and with a second projection **138** adapted to engage corresponding projections **140**, **141** of the arms of the sensing rollers **126**, **128** respectively, whereby when at least one sensing roller is at rest (that is, does not lie in the card path the corresponding projection **138**, keeps the arm **135** rotated in the clockwise direction, because the spring **130**, **132** respectively, overcomes the spring **139**.

On the contrary, when both the sensing rollers are raised by the card, the arm **135** is free to rotate counter-clockwise, whereby its projection **136** is allowed to actuate the switch **137**.

The card **150**, having been manually introduced into the inlet opening **113**, is engaged by the first pair of rollers continously rotating **116**, **118** and pushed toward the second pair of continuously rotating rollers **117**, **119**, which causes the card to advance past the magnetic head **129** at substantially constant speed. The first sensing roller **126** is raised upon been reached by the leading edge of the card. However, as the second sensing roller remains in the rest position, the arm **135** remains in its clockwise position, whereby the projection **136** is prevented from actuating switch **137** until the second sensing roller **128**, upon being reached in turn by said leading edge, is raised.

When, thereafter, the trailing edge of the card reaches the first sensing roller **126**, the arm **125** rotates clockwise pushing the arm **135** in the same direction, whereby the switch **137** is disengaged. Therefore the switch **137** is adapted to generate an electric signal **A0** beginning when the leading edge of the card reaches the second sensing roller **128** and ending when the trailing edge of the card reaches the first sensing roller **126**, so as to identify the time interval during which the good part of the track **151** passes under the mangetic head **129**.

At the end of its path the card **150** is disengaged from the pair of rollers **117** and **119**, whereby it is stopped by friction in such a position that its leading edge projects out of the outlet opening **144** so as to allow manual extraction. In this end position a predetermined portion of the card adapted to bear visible designations pertaining to the information recorded in code form thereon lies under an opening **142** of the computer cover, in front of the subroutine keys **V1**, **V2**, **V3** and **V4**.

More particularly each card may bear on the position facing a subroutine key a brief written statement or symbol of the operation performed by the computer under the control of the program subroutine which in the program is headed by the reference instruction having the same code of said key. Therefore, with reference to the previously cited example, the subroutine key will be labelled "sine" and "cosine" when the first card and the second card respectively is inserted in the computer.

The keyboard **100**, the printing unit **103** and the card handling unit are three independent mechanical groups fixed to the frame **148** which may be counter-clockwise rotated (FIG. 9) around an axis **143**, whereby all the mechanical parts of the computer may be raised as block for inspection and maintenance.

According to an embodiment of the invention, the card **150** is provided with a single magnetic track **151** for storing the entire contents of five registers of the memory LDR.

On the track **151**, the eight binary denominations of each character are followed by four blank denominations, whereby each character, as recorded on the card, comprises twelve denominations.

Therefore, assuming each memory register containing

24 digits, the track **151** contains an uninterrupted series of $12 \cdot 24 \cdot 5 = 1440$ binary denominations, among which only 960 contain bits to be transferred into the memory registers.

The card **150**, having been introduced manually in the inlet opening **113**, advances at constant speed past the magnetic head **129**, whereby the 1440 binary denominations of the mangetic track **151** are scanned at a constant frequency in the same direction both when reading and when recording.

When reading the card, each group of eight bits read on the card and representing a character is stored in the shift register K. While the magnetic head scans the four next following blank denominations, said eight bits are transferred from the register K to the memory register at present addressed.

Similarly, when recording on the card, while the magnetic head scans a group of four blank binary denominations, a character is transferred from the memory register at present addressed to the register K; when, thereafter, the magnetic head scans the eight next following binary denominations, said character is extracted from the register K and recorded on the card.

More particularly, according to an embodiment of the invention the card moves at such a speed, that its successive binary denominations are scanned at 0.6 millisecond intervals, a memory cycle being 2.1 milliseconds long, whereby the time consumed in scanning the four blank denominations is sufficient for having access to whatsoever decimal denomination of the delay line in order to enter therein or extract therefrom a certain character. Therefore it is apparent that the blank space separating two contiguous characters on the card corresponds to a time interval greater than the access time of the delay line memory, whereby the successive characters may be sequentially transferred to and from the card, serially by bit, through a buffer storage (register K) having a single character capacity, thus substantially reducing the cost of the apparatus.

According to another feature of the invention in each group of four blank binary denominations of the card at least one is used to store check bits associated with the character recorded in the eight contiguous binary denominations, said check bits being computed when recording the card and used and destroyed when reading the card.

Moreover all the binary denominations of the card, included the blank denominations, are counted when scanning the card to ascertain that none has been skipped or read more than once.

FIGS. 11*a* and 11*b* show some parts of the circuits of the computer involved in the card processing operation.

The normally open switch **137** is closed when the card **150** engages both sensing rollers **126** and **128**, so as to energize one input of both gates **218** and **219** (FIG. 11*a*).

A commutator **205** is manually actuated by the operator for preselecting either the card reading operation or the card recording operation by energizing the other input of gate **218** or **219** respectively.

Therefore while reading and recording on the card the terminal AL, AS respectively, generates a signal lasting the entire time interval spent by the magnetic head **129** in scanning the track **151**.

The magnetic head **129** is connected to a reading-recording amplifier **206**.

According to an embodiment of the invention, the magnetic flux in the magnetic track **151** exhibits (line NL of FIG. 12) a series of reversals or transitions, called clock flux transitions, spaced a distance corresponding to 600 microseconds, the zone between two contiguous clock flux transitions constituting a binary denomination of the card. Each bit "1" or "0" is represented by the presence and the absence respectively, of a flux transition, called information flux transition, spaced a distance corresponding to 200 microseconds from the clock flux transition which marks the beginning of the corresponding binary denomination. Said flux distribution is produced by a signal having a similar wave form and fed to the input **207** of the amplifier **206** via a gate **209** from a bistable device NL, which when recording is fed with the binary signals issued by the register K and serves the purpose of shaping said signals as required for modulating the magnetic flux. On the output **208** of the amplifier **206** a short pulse LS is obtained upon reading each clock flux transition, and each information flux transition. The signals LS produced by sensing the information flux transitions, having been discriminated by a gate **228** and reshaped by a bistable device NH, are fed into the register K via the gate **230**.

An oscillator OR, which is operative only when the signal AS indicating that the track **151** is being scanned for recording is present on its input, produces on its output a train of pulses OR (FIG. 12), each one 200 microseconds long and having a 600 microseconds repetition period. Moreover, through differentiating circuits **211** and **212** the oscillator OR produces a short pulse ORF, ORC respectively, at the leading edge, trailing edge respectively, of each pulse OR.

Each pulse ORF starts a monostable circuit OS having a 400 microseconds inherent delay, whereby the monostable circuit OS produces on its output a series of pulses, each one 400 microseconds long, at 600 microseconds intervals. Moreover, via differentiating circuits **214** and **213** a short pulse OSF, OSC respectively, is produced at the leading edge, trailing edge respectively, of each pulse OS.

On the contrary, when signal AL is present to indicate that the track **151** is being scanned for reading, the oscillator OR is inoperative, and the monostable circuit OS is started via a gate **215** by each signal produced by the amplifier **206** upon reading a clock flux transition.

The pulses OSF are used as counting pulses for advancing a modulo-twelve counter **216**, so as to energize an output **H1–8** when the magnetic head scans the first eight binary denominations of each character on the card, the output **119** when scanning the ninth binary denomination of each character on the card and the output $\overline{\text{H12}}$ when scanning all denominations but the twelfth (last) denomination of each character.

Both when reading and when recording, the pulses OSC are used as shift pulses for the register K, whereby upon receiving a pulse OSC on the input **4** via gate **217** the contents of the register K are shifted one binary denomination leftward.

Therefore it is apparent that when reading the card, the bits are shifted in the shift register K in synchronism with the scanning of the card, since the monostable circuit OS is then fed with the signals produced by the reading amplifier **206**, and that when recording the bits are shifted in the shift register K in synchronism with the scanning of the card because the recording process is timed by the oscillator OR which also controls the monostable circuit OS.

The input **13** of the register K is connected via gate **221** to the output LI, LJ, LZ, LD, LE of the register I, J, Z, D, E of the memory LDR respectively when recording. Similarly, when reading the card the output **14** of the register K is connected via gate **231** to the input SI, SJ, SZ, SD, SE of said registers respectively.

Said registers are addressed by means of gates **200, 201, 202, 203, 204,** and **234, 335, 236, 237, 238.**

The operation of the computer when recording on a card will be now briefly described.

When the commutator **205** is preset in the "recording" position, so as to produce the signal AS**0**, the leading edge of this signal energizes the bistable device A7 (FIG. 11*b*), which serves the purpose of indicating that from that moment a character may be transferred from the memory LDR to the register K. Upon completion of this transfer operation, the bistable device A7 is deenergized,

for preventing the other characters from being unduly transferred.

The first character transferred is the character stored in the first decimal denomination of the register J. The trailing edge of the signal A**10** (stop oscillator **44**) via gate **220** energizes the bistable device A**9**, which is thereafter deenergized by the next following clock pulse T**1**, which in this case occurs in the first bit period of the first digit period of the new memory cycle. Said pulse T**1** energizes the bistable device A**3**, which thereafter remains energized during the entire first digit period, for indicating that in said digit period the character to be transferred is delivered on the output of the delay line.

More particularly, the bistable device A**3** when energized opens gate **221**, whereby the eight bits of the first character extracted from the delay line via the gate **235** are transferred into the register K, and further opens gate **222**, whereby the register K receives a train of eight shift pulses M**4**, one in each bit period, with the frequency of the signals in the delay line. Therefore said eight bits are shifted into the register K and thereafter stored therein until recording on the card. After said digit period, the bistable device A**3** is deenergized by the clock pulse T**1**, thus deenergizing also the bistable device A**7**. During the digit period identified by the bistable device A**3** being energized, in the tag-bit control circuit a tag bit $B1M$="1" is written into the register M via gate **225**. Said tag bit is adapted to thereafter indicate, what is the character last transferred from the delay line LDR to the register K.

In the meantime the operator inserts the card into the computer, whereby when the track **151** begins to be scanned by the head **129**, the switch **137** generates the signal AS.

The presence of this signal makes the oscillator operative. The first pulse OSF produced by the oscillator OS advances the counter **216** so as to energize its output **H1–8** and switches the bistable device NL, whereby the amplifier **206** records on the card the first flux reversal, that is the clock flux transition marking the beginning of the first binary denomination 200 microseconds later, the oscillator OR produces a first signal OSC which via gate **223** either energizes the counting input **210** of the bistable device NL or not depending on whether the first bit of the character, being now retained in the output stage **K1** of the register K, has the value 1 or 0. In FIG. 12 the first and second characters have been assumed to be 1, 0, 0, 1, 0, 1, 0, 0 and 0, 1, 0, 0, 0, 1, 0, 0 respectively. Via gate **209** and amplifier **206**, the output signal of the bistable device NL is recorded on the card. 200 microseconds later, the oscillator OR produces a first signal OSC, which through gate **217** causes the contents of the register K to be shifted one stage, whereby the second bit of the character to be recorded on the card is shifted into the output stage **K1**.

200 microseconds later, the oscillator OR produces a second pulse OSF, which steps the counter **216** and switches the bistable device NL, whereby the second clock flux transition is recorded on the card. 200 microseconds later, the oscillator OR produces a second pulse ORC, which through gate **223** causes the bistable device NL either to switch or not depending on whether the bit now retained in the output stage **K1** is 1 or 0. In FIG. 4 this bit is 0. 200 microseconds later, the oscillator OR produces a second pulse OSC, which through gate **217** shifts the contents of the register K, whereby the third bit is shifted into the output stage **K1**. Said third bit and the following five bits are similarly recorded on the card.

The ninth pulse OSF deenergizes the output **H1–8** of the counter **216** and energizes the output **H9**.

The signal **H1–8** being absent, gates **217** and **223** are closed, register K receives no more shift pulses from the oscillator OR and the connection of its output **14** to the magnetic head **129** is interrupted.

The trailing edge of signal **H1–8** via gate **224** energizes the bistable device A**7**. Therefore the reading signal LB**1M**, produced upon reading the tag bit B**1M** out of the delay line, is allowed to energize the bistable device A**9** via gate **226**. Bistable device A**9** when energized identifies the digit period next preceding the character to be transferred from the delay line LDR to the card, moreover it causes the bistable device A**3** to be energized for identifying the digit period in which the character to be recorded on the card is read out of the delay line.

The bistable device A**3** being energized opens gates **221** and **222**, whereby during only said digit period the memory LDR is connected to the register K, which in turn receives eight shift pulses M**4** at the frequency of the pulses in the delay line.

Therefore the second character of the register J is transferred into the register K. In the meantime the oscillator OR remains operative, whereby the ninth pulse ORC causes the bistable device NL to be either switched or not via gate **227** depending on whether said bistable device NL is energized or not, whereby a new flux transition is either recorded or not on the card to make the total number of transitions recorded in the first nine denominations equal to an even number. Otherwise stated, said new flux transition represents a parity bit.

On the contrary, in the following (tenth, eleventh, twelfth) denominations, only the clock flux transition is recorded.

The thirteenth pulse OSF reenergizes the output **H1–8** of the counter **216**, whereby gates **223** and **217** are opened again in order to connect the register K to the magnetic head and to shift the second character out of the register K to the card itself.

The following characters are recorded in a similar manner.

The card reading operation will now be briefly described (FIG. 12*b*).

Upon introducing the card in the computer, the first clock flux transition is sensed and produces a reading signal LS which starts the monostable circuit OS via gate **215**. Therefore a first pulse OSF is produced, whereby counter **216** is stepped to energize output **H1–8**. This causes gate **217** to be opened for feeding the register K with a train of eight shift pulses OSC having a frequency controlled by the clock flux transitions recorded on the card.

The monostable circuit remains energized 400 microseconds, whereby during this interval the reading signal LS representing the first bit is fed via gate **228** to energize the bistable device NH, whose output will thus represent said bit being now read on the card.

The output signal of the bistable device NH is fed via gate **230** to register K, whereby, upon receiving the first shift pulse OSC via gate **217**, said bit having been read on the card is transferred into the stage **K8**. About 200 microseconds later, the second clock flux transition is read on the card, whereby a signal LS starts the monostable circuit OS again. Therefore a second signal OSF is produced for stopping the counter **216** and resetting the bistable device NL.

Moreover the signal OS by opening the gate **228** discriminates the time interval during which the information flux transition representing the second bit may occur. Said second bit is therefore staticized in the bistable device NH and then transferred into stage *Ku*. In a similar manner the following six bits of the first character are read on the card. Upon reading the ninth clock flux transition, the ninth pulse OSF causes the counter **216** to advance so as to energize output **H9** and deenergize output **H1–8**. Therefore the gate **217** is closed for preventing the register K from being fed with shift pulses having the frequency of the signals read on the card.

The trailing edge of the signal **H1–8** energizes via gate **224** the bistable device A**7** to indicate that at present the register K must be connected to the delay line LDR

for transferring the first character into the register J. Said trailing edge may occur at any point of a memory cycle. At the end of this cycle, the bistable device **A9** is energized via gate **220** in the manner previously explained for the recording operation, whereby at the beginning of the next following memory cycle (beginning of the first digit period C1) the bistable device **A3** is energized to identify the digit period C1 as the digit period in which the character is to be transferred.

More particularly the bistable device **1A3** being energized opens gates **231** and **222** to connect the register K to the memory LDR and to feed the register K with a train of eight shift pulses **M4** synchronized with the pulses in the delay line, whereby the first character is written into the first denomination of the register J.

In the card reading phase, the bistable device NL receives every signal OSF produced upon sensing a clock flux transition and every signal issued by the gate **228** upon sensing an information flux transition.

Therefore the bistable device NL when reading the card gives a replica of the signal which had been fed to the input **207** of the amplifier **206** when recording. When scanning the end of the ninth binary denomination of the card (signal H9 is present, signal OS is absent) the bistable device NL must be energized, because it must have made nine non significant commutations and an even number of significant commutations. If on the contrary the bistable device remains then deenergized, the output of a gate **232** is energized to provide an error signal ERL.

In a similar manner the following characters are read on the card.

At the end of the reading operation, when the signal AL has disappeared, counter **216** must have its output **H12** deenergized, since a multiple of twelve denominations should have been scanned on the card.

If this condition does not occur, the output of a gate **233** is energized to produce an error signal ERL.

As shown in FIG. 11*a*, when reading the card, the output of the shift register K is connected to the input of the register I, J, Z, D, E via gate **200**, **201**, **202**, **203** and **204** respectively when reading the first, second, third, fourth, fifth group of 24 characters recorded on the card respectively.

For this purpose said five gates are sequentially opened by address signals produced by the address decoder **17**. According to an embodiment of the invention, the instruction staticisor **16** is also used as address register for sequentially addressing said five registers in the card reading phase.

As shown in FIG. 11*a*, in this phase (signal AL is present) the register I and J, which cannot be addressed by the address signals **Y1** to **Y8** issued by the decoder **17** and allotted to the normally addressable register M, N, R, Q, U, Z, D, E, are addressed by the address signals $Y1 \cdot AL$ and $Y2 \cdot AL$ respectively.

Since the registers I, J, Z, D, E involved in the card reading operation must be addressed sequentially, means must be provided for causing the address decoder **17** to sequentially generate the corresponding address signals **Y1**, **Y2**, **Y6**, **Y7**, **Y8**. To this end the instruction staticisor **16** is conditionable by the signal AL (card reading phase) to act as a counter having suitable internal feedback connections for generating said sequence of address signals upon receiving successive counting pulses. Alternatively, the code representation of said addresses may be selected in such a way that by entering a certain group of bits into the instruction staticisor **16** working as a shift register and then shifting said bits, the successive address signals are generated.

Each counting pulse for stepping the decoder **17** from an address to the next following address is generated when filling up a register with the characters read on the card is completed.

More particularly, when reading on the card the last (24th) character to be entered in the register J, the tag bit **B1M** (which is shifted along the delay line to mark the denomination in which the next following character must be entered) will be in the last decimal denomination. This means that the register J has been filled up and that register **1** may be subsequently addressed. As previously explained, the bistable device **A22** is energized during the last digit period of each memory cycle. Therefore a signal indicating the coincidence of the signals **A22** and **A3** is used as a counting signal for stepping said instruction staticisor to generate the address of said next following register J.

It is thus apparent that the instant in which the next following register must be addressed is determined without counting the number of transferred characters, whereby an expensive character counter is dispensed for.

In a similar manner the memory registers are addressed when recording on the card.

According to an embodiment of the present computer (FIGS. 9 and 10), the keyboard **101**, comprises a slide **160** for each key having code slots corresponding to the code of said key. Upon depressing said key, the corresponding code slide **160** is moved by the motor **120** rightward (FIG. 9), whereby seven code bars **161** to **167** are positioned according to said code. Each code bar in turn causes a separate code slide similar to the slides **160** to be moved rightward so as to actuate a corresponding switch **102**. Therefore the code bars **161**, **162**, **163**, **164** produce four binary signals representing the depressed key. The code bars **165** and **166** produce a pair of signals which are combined to obtain the signals G1, G2 and G3 indicating whether the numeric keyboard **65** or the address keyboard **68** or the function keyboard **69** has been operated. The code bar **167** provides a strobe signal for the computer upon actuating any key. Moreover the slides **160** associated with some keys, for instance the minus key and the decimal point key, are adapted to directly actuate a corresponding separate switch **102**.

The serial printing unit **103** comprises a fixed type drum **104** and a traveling printing hammer **105** for printing on a paper roll **106**.

The back part of the computer comprises the electronic circuit modules **107** mounted on printed circuit cards **108** interconnected by means of edge connectors **109** and printed circuit cards **110**. A box **112** contains the delay line LDR.

We claim:

1. A program controlled electronic computer comprising:
- (a) means for storing a program comprising a series of instructions including normal instructions and jump instructions, and also comprising marks for designating program subroutines, each one of said jump instructions corresponding to at least one of said marks,
- (b) means for selecting one of said instructions,
- (c) means responsive to said selecting means upon selecting one of said normal instructions for controlling said computer to perform predetermined operations,
- (d) means effective upon completion of said operations for conditioning said selecting means to select the next following instruction in said series, and
- (e) means responsive to said selecting means upon selecting one of said jump instructions for sequentially searching said program and for conditioning said selecting means, upon finding a mark corresponding to said selected jump instruction, to select the first instruction of the corresponding subroutine.

2. A program controlled electronic computer comprising:
- (a) means for storing a program comprising a series of instructions including normal instructions, jump instructions and reference instructions, each one of said reference instructions heading a program subroutine, and each one of said jump instructions corresponding to at least one of said reference instructions,
- (b) means for selecting one of said instructions,
- (c) means responsive to said selecting means upon selecting one of said normal instructions for controlling said computer to perform predetermined operations,
- (d) means effective upon completion of said operations for conditioning said selecting means to select the instruction next following said selected normal instruction in said series, and
- (e) means responsive to said selecting means upon selecting one of said jump instructions for sequentially searching the instructions of said series and for conditioning said selecting means, upon finding a reference instruction corresponding to said selected jump instruction, to select the instruction next following said selected jump instruction in said series.

3. A program controlled electronic computer comprising:
- (a) means for storing a program comprising a series of instructions,
- (b) an instruction register,
- (c) means controlled by said stored program for transferring a predetermined instruction from said program storing means to said instruction register,
- (d) means automatically operative upon entering an instruction into said instruction register for executing said last mentioned instruction, and
- (e) a set of control keys for entering an instruction into said instruction register.

4. A program controlled electronic computer comprising:
- (a) means for storing a program comprising a series of instructions,
- (b) means for indicating an instruction,
- (c) means controlled by said stored program for transferring a predetermined instruction from said program storing means to said instruction indicating means,
- (d) means responsive to said indicating means for executing said indicated instruction, and
- (e) control keys conditionable for entering an instruction into either said indicating means or said program storage means.

5. A program controlled electronic computer comprising:
- (a) means for storing a program comprising a series of instructions including normal instructions and jump instructions, each one of said jump instructions designating a corresponding subroutine in said stored program,
- (b) means for indicating an instruction,
- (c) means responsive to said indicating means upon indicating one of said normal instructions for controlling said computer to perform predetermined operations,
- (d) means effective upon completion of said operations for conditioning said indicating means to indicate the instruction next following said normal instruction in said series,
- (e) means responsive to said indicating means upon indicating one of said jump instructions for controlling said computer to execute the subroutine designated by said jump instruction, and
- (f) subroutine keys operable for causing said indicating means to indicate predetermined ones of said jump instructions respectively.

6. A program controlled electronic computer comprising:
- (a) means for storing a program comprising a series of instructions,
- (b) an instruction register,
- (c) sequence control means,
- (d) means controlled by said sequence control means for executing an instruction contained in said register,
- (e) means controlled by said sequence control means and operative upon completion of the execution of said instruction for transferring a predetermined instruction from said program storing means to said instruction register,
- (f) and a set of control keys for manually entering an instruction into said instruction register.

7. A program controlled electronic computer comprising
- (a) storage means,
- (b) a plurality of subroutine keys,
- (c) means for reading and entering into said storage means at least one subroutine recorded on a record member, each subroutine having associated therewith a designator, each designator corresponding to a separate one of said subroutine keys, the actuation of a subroutine key being effective to select from said storage means the entered subroutine having associated therewith the corresponding designator and
- (d) means responsive to the actuation of a subroutine key for causing the computer to automatically execute the selected corresponding subroutine.

8. An electronic computer controlled by a program including a plurality of subroutines, comprising:
- (a) a plurality of subroutine keys, each one corresponding to one of the subroutines of said program;
- (b) means for receiving a record member;
- (c) means for reading the record member entered into said receiving means;
- (d) said record member having recorded thereon said program, and having portions bearing visible designations of the subroutines of said recorded program respectively;
- (e) means fed by said reading means for storing said program;
- (f) means for holding said entered record member with said portions lying in visual correspondence with the corresponding subroutine keys respectively;
- (g) and means responsive to actuation of said subroutine key for causing the computer to automatically execute the corresponding stored program subroutine.

9. A generally desk-top size program controlled electronic computer comprising:
- (a) memory means having a plurality of addressable locations for storing both data and a program comprising a series of instructions for operating the computer;
- (b) a plurality of record members each having recorded thereon a program and each having associated therewith a visible designation for indicating the program recorded thereon;
- (c) record processing means including:
  - (1) record receiving means, and
  - (2) means settable for automatically reading by scanning an individual one of said record members inserted into said record receiving means in a single scanning operation and for entering the program so read into a portion of said memory means with programs from successive record members superimposed on instructions previously stored in said portion of said memory means;
- (d) means for serially reading said entered instructions from said portion of said memory means, and
- (e) means responsive to the instructions read from said memory means for selectively performing arithmetic and logical operations on the data stored in selected locations in said memory means.

10. The program controlled electronic computer of claim 9 further comprising:

(a) a keyboard including a set of control keys for controlling said computer and

(b) means for drawing said received card along a curved path passing below said keyboard, the entrance and exit of said path being substantially coplanar with said keyboard.

11. The apparatus of claim 9, further including control keys operable when actuated to record different instructions into said memory means, said said processing device being operable to record onto a record card inserted into its receiving means a series of instructions recorded in said memory means.

12. A program controlled electronic computer comprising:

(a) a storage for storing at least one series of instructions,

(b) control keys for entering instructions into said storage,

(c) means for selectively reading said instructions from said storage for controlling the operation of said computer, and

(d) record processing means including:

(1) means for receiving individual manually insertable record members,

(2) read-write means for transferring information between said storage and said record members and,

(3) means settable by an operator for selectively controlling the direction of the transfer of said information.

13. In a data processing apparatus:

(a) a storage having an access time,

(b) a buffer memory having a storage capacity of a predetermined number of bits,

(c) means for scanning individual record members bearing at least one track of digitally coded information arranged in groups of bits, each of said bits of a group being separated from adjacent bits by a distance corresponding to an interval between scanning adjacent bits which is less than said access time, each said group containing a number of bits whose maximum is equal to the capacity of said buffer memory, adjacent groups being separated by a distance corresponding to an interval between scanning the last bit of one group and the first bit of the next group which is at least as great as said access time, and

(d) means for writing the bits of each group of said scanned information into said buffer memory while that group is being scanned and for transferring each group to said storage during scanning of the distance separating adjacent groups.

14. The apparatus of claim 13 further comprising checking means associated with said scanning means for counting said scanned bits including bits recorded between said groups.

15. The apparatus of claim 13 having check bits recorded between said groups for checking the contiguous groups.

16. A generally desk-top size program controlled electronic computer comprising:

(a) memory means having addressable locations for storing data to be processed and a program comprising a series of instructions for controlling the operation of said computer;

(b) record processing means, including:

(1) record receiving means; and

(2) means settable for automatically reading by scanning in a single scanning operation an individual manually insertable record member containing a program inserted into said record receiving means and for entering the program so read into a portion of said memory means, with programs from successive record members superimposed on instructions previously stored in said portion of said memory means;

(c) means for serially reading said instructions from said portion of said memory means; and

(d) means responsive to the reading of said instructions from said memory means for selectively performing arithmetic and logical operations on the data stored in selected locations of said memory means.

References Cited

UNITED STATES PATENTS

2,883,106	4/1959	Cornwell et al.	235—61.6
3,031,136	4/1962	Blumenthal et al.	235—61.11
3,037,698	6/1962	Saxenmeyer	235—153
3,145,920	8/1964	Berlinsky et al.	234—55
3,150,351	9/1964	Tackovich	340—172.5
3,187,321	6/1965	Kameny	340—345
3,292,155	12/1966	Neilson	340—172.5
3,355,714	11/1967	Culler	340—172.5
3,201,759	8/1965	Kelly	340—172.5
3,335,265	8/1967	Apfelbaum	235—61.11
3,353,163	11/1967	Soule et al.	340—172.5
3,360,781	12/1967	Boehnke	340—172.5

OTHER REFERENCES

Buchholz: "The System Design of the IBM 701 Computer," Proceedings of the I.R.E., October 1953, pp. 1262–1275.

Design Features of the "Jaincomp-c and Jaincomp-d Electronic Digital Computers," Convention Record of the I.R.E., vol. 2, 1954, part 4 (Electronic Computer), pp. 98–104.

RAULFE B. ZACHE, Primary Examiner

U.S. Cl. X.R.

235—61.6

Disclaimer

3,495,222.—*Pier Giorgio Perotto*, Turin, and *Giovanni De Sandre*, Sacile, Italy. PROGRAM CONTROLLED ELECTRONIC COMPUTER. Patent dated Feb. 10, 1970. Disclaimer filed Mar. 17, 1971, by the assignee, *Ing. C. Olivetti & C., S.p.A.*

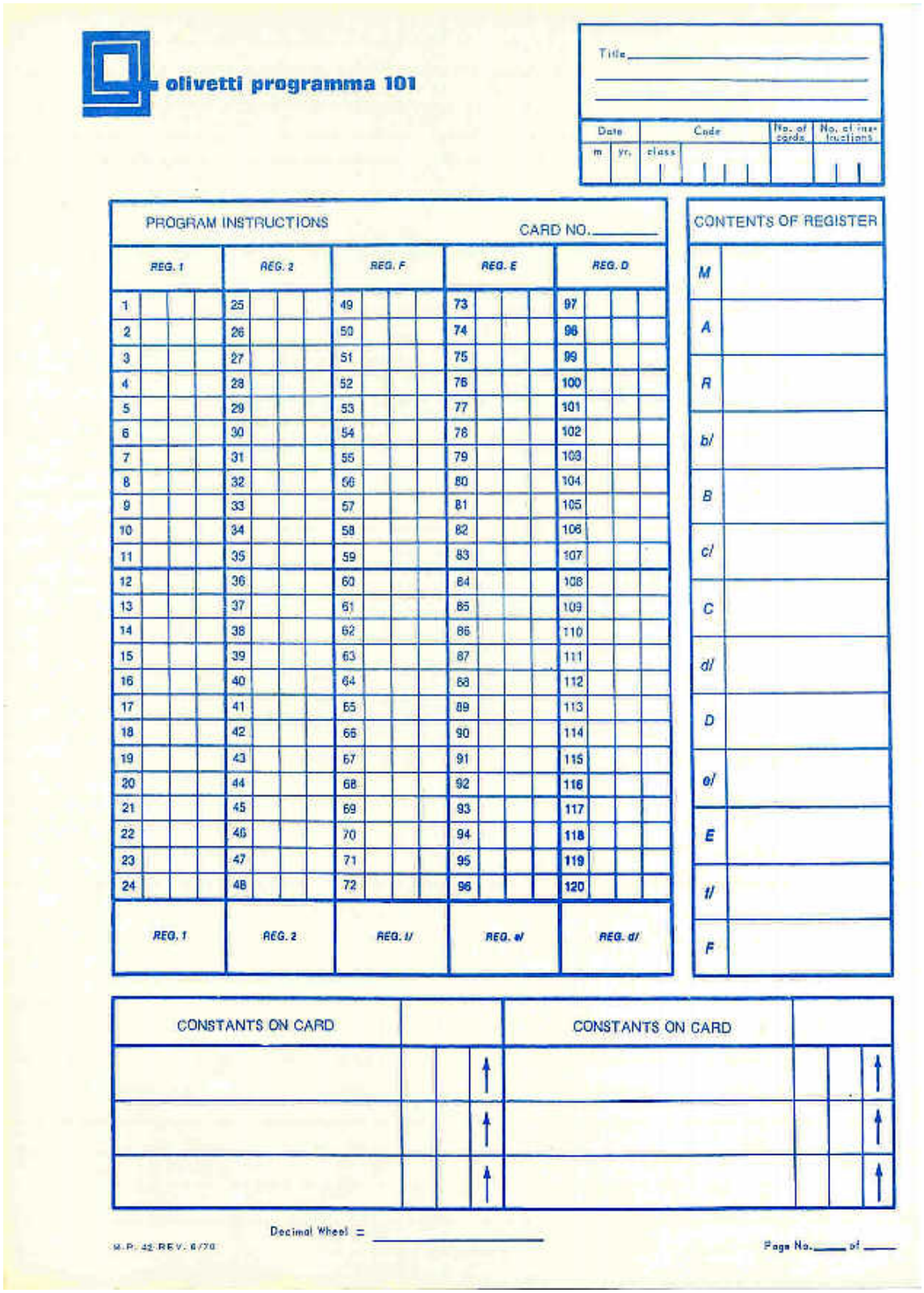
Hereby enters this disclaimer to claims 3, 4 and 6 of said patent.

[*Official Gazette June 29, 1971.*]

APÉNDICE C

Plantilla de programación

Ya que la Programma 101 disponía de un lenguaje ensamblador fácil de usar, Olivetti diseñó un formulario —la que se muestra en este apéndice— para que sirviera de plantilla a aquellos usuarios que se dedicaran a programar esta computadora. Dicha plantilla muestra de forma compacta toda la información necesaria para la función mencionada, tal como el contenido de los registros, tanto el código del programa como constantes necesarias, el estado de la rueda que configura el número de decimales, así como información para su archivo y asociación con otras tarjetas magnéticas donde se almacenara dicha programación.



olivetti programma 101

Title

Date		Code		No. of cards	No. of instructions
m	yr.	class			

PROGRAM INSTRUCTIONS CARD NO.

REG. 1	REG. 2	REG. F	REG. E	REG. D
1	25	49	73	97
2	26	50	74	98
3	27	51	75	99
4	28	52	76	100
5	29	53	77	101
6	30	54	78	102
7	31	55	79	103
8	32	56	80	104
9	33	57	81	105
10	34	58	82	106
11	35	59	83	107
12	36	60	84	108
13	37	61	85	109
14	38	62	86	110
15	39	63	87	111
16	40	64	88	112
17	41	65	89	113
18	42	66	90	114
19	43	67	91	115
20	44	68	92	116
21	45	69	93	117
22	46	70	94	118
23	47	71	95	119
24	48	72	96	120
REG. 1	REG. 2	REG. 1/	REG. e/	REG. d/

CONTENTS OF REGISTER

M	
A	
R	
b/	
B	
c/	
C	
d/	
D	
e/	
E	
f/	
F	

CONSTANTS ON CARD		CONSTANTS ON CARD	

Decimal Wheel =

M.P. 43 REV. 6/70

Page No. of

APÉNDICE D

Ejemplos de código

En este apéndice, se muestra el código de cada uno de los algoritmos vistos en el capítulo 5, implementados en la plantilla de programación que se ha podido observar en el apéndice anterior, tal y como lo harían los programadores de la Programma 101 en los tiempos en que esta era utilizada.

olivetti programma 101

Title FACTORIAL

Date m	Date yr.	Code class	Code	No. of cards	No. of instructions
				1	16

PROGRAM INSTRUCTIONS — CARD NO. 1/1

	REG. 1	REG. 2	REG. F	REG. E	REG. D
1	A V	25	49	73	97
2	S	26	50	74	98
3	D ↑	27	51	75	99
4	↓	28	52	76	100
5	A W	29	53	77	101
6	A / ↑	30	54	78	102
7	D / ↓	31	55	79	103
8	-	32	56	80	104
9	/ V	33	57	81	105
10	D ◇	34	58	82	106
11	V	35	59	83	107
12	A / V	36	60	84	108
13	D ↕	37	61	85	109
14	D x	38	62	86	110
15	D ↕	39	63	87	111
16	W	40	64	88	112
17		41	65	89	113
18		42	66	90	114
19		43	67	91	115
20		44	68	92	116
21		45	69	93	117
22		46	70	94	118
23		47	71	95	119
24		48	72	96	120
	REG. 1	REG. 2	REG. //	REG. W	REG. d/

CONTENTS OF REGISTER

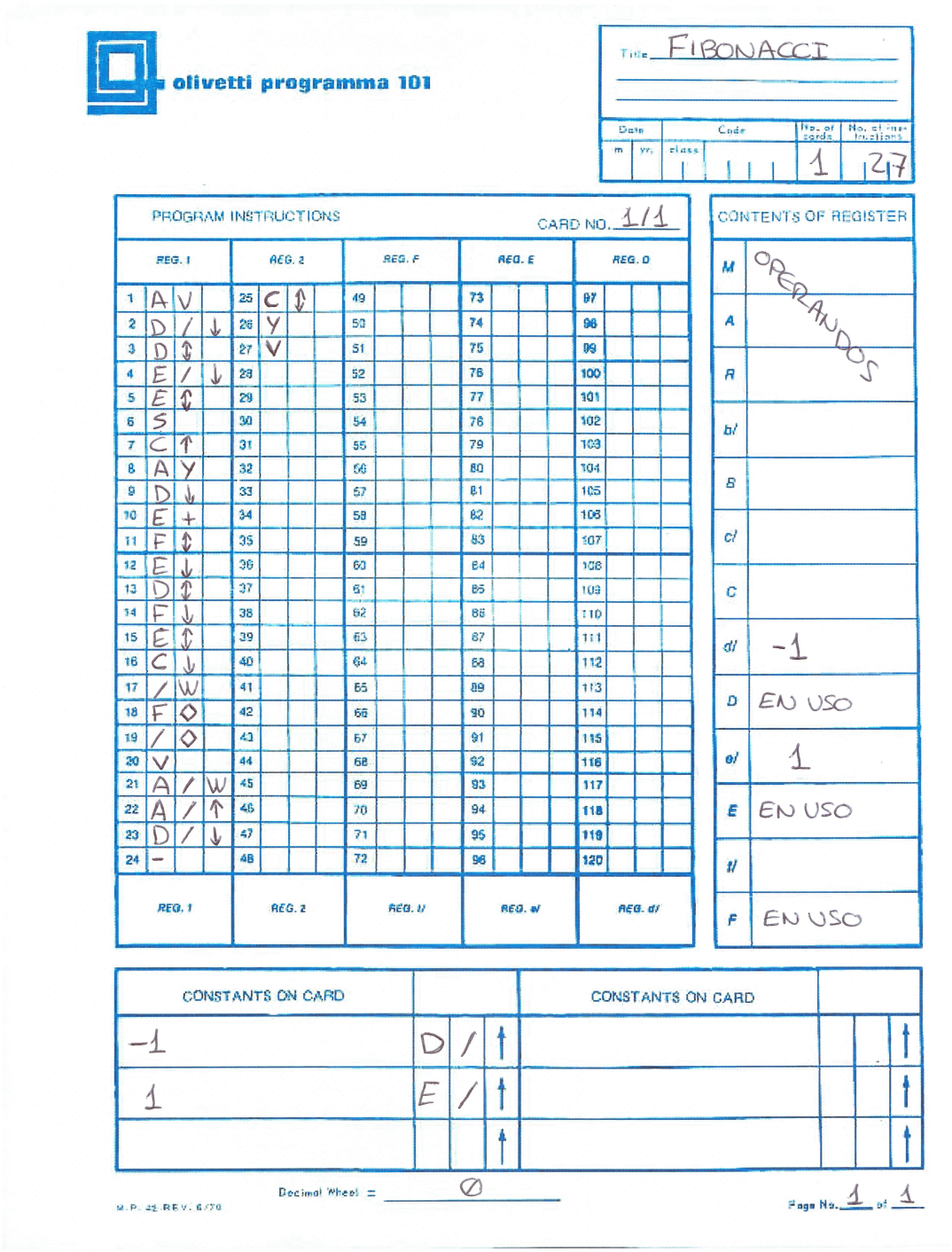
M	OPERANDOS
A	
R	
b/	
B	
c/	
C	
d/	
D	EN USO
e/	
E	
f/	
F	

CONSTANTS ON CARD		CONSTANTS ON CARD	
	↑		↑
	↑		↑
	↑		↑

Decimal Wheel = 0

M.P. 45 REV. 6/70

Page No. 1 of 1



olivetti programma 101

Title: FIBONACCI

Date m	Date yr.	Code class	Code	No. of cards	No. of instructions
				1	27

PROGRAM INSTRUCTIONS — CARD NO. 1/1

	REG. 1	REG. 2	REG. F	REG. E	REG. D
1	A V	25 C ↕	49	73	97
2	D / ↓	26 Y	50	74	98
3	D ↕	27 V	51	75	99
4	E / ↓	28	52	76	100
5	E ↕	29	53	77	101
6	S	30	54	78	102
7	C ↑	31	55	79	103
8	A Y	32	56	80	104
9	D ↓	33	57	81	105
10	E +	34	58	82	106
11	F ↕	35	59	83	107
12	E ↓	36	60	84	108
13	D ↕	37	61	85	109
14	F ↓	38	62	86	110
15	E ↕	39	63	87	111
16	C ↓	40	64	88	112
17	/ W	41	65	89	113
18	F ◇	42	66	90	114
19	/ ◇	43	67	91	115
20	V	44	68	92	116
21	A / W	45	69	93	117
22	A / ↑	46	70	94	118
23	D / ↓	47	71	95	119
24	−	48	72	96	120
	REG. 1	REG. 2	REG. M	REG. M	REG. M

CONTENTS OF REGISTER

M	ORDENADOS
A	
R	
b/	
B	
c/	
C	
d/	-1
D	EN USO
e/	1
E	EN USO
f/	
F	EN USO

CONSTANTS ON CARD		CONSTANTS ON CARD	
-1	D / ↑		↑
1	E / ↑		↑
	↑		↑

Decimal Wheel = ∅

M.P. 43 REV. 6/70

Page No. 1 of 1

olivetti programma 101

Title: ECUACIÓN DE SEGUNDO GRADO

Date m	Date yr.	Code class	Code	No. of cards	No. of instructions
				1	416

PROGRAM INSTRUCTIONS — CARD NO. 1/1

	REG. 1	REG. 2	REG. F	REG. E	REG. D
1	A V	25 A / Z	49	73	97
2	S	26 A -	50	74	98
3	B ↑	27 -	51	75	99
4	S	28 C ◇	52	76	100
5	C ↑	29 A ⌐	53	77	101
6	S	30 A ◇	54	78	102
7	↕	31 C ◇	55	79	103
8	B W	32 -	56	80	104
9	B ÷	33 A ◇	57	81	105
10	B ↕	34 V	58	82	106
11	A +	35 A / V	59	83	107
12	C ↕	36 A ⌐	60	84	108
13	C ÷	37 C ↕	61	85	109
14	A -	38 C +	62	86	110
15	-	39 A ◇	63	87	111
16	A X	40 C -	64	88	112
17	C ↑	41 -	65	89	113
18	B -	42 A ◇	66	90	114
19	/ V	43 V	67	91	115
20	A -	44 A W	68	92	116
21	-	45 C / ↓	69	93	117
22	/ Z	46 C W	70	94	118
23	C ◇	47	71	95	119
24	V	48	72	96	120
	REG. 1	REG. 2	REG. U	REG. M	REG. D

CONTENTS OF REGISTER

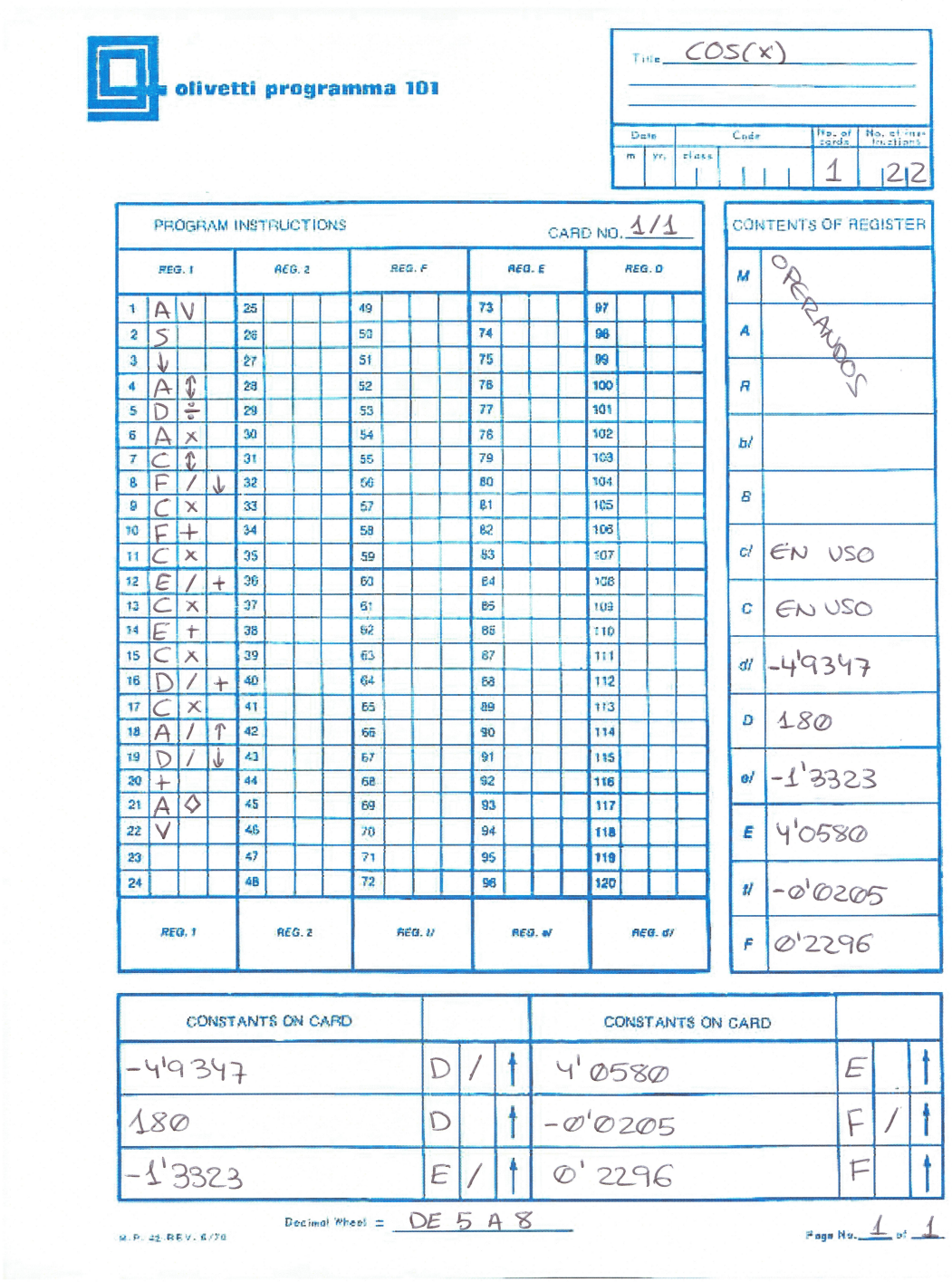
M	OPERANDOS
A	
R	
b/	
B	EN USO
c/	
C	EN USO
d/	
D	
e/	
E	
I/	
F	

CONSTANTS ON CARD		CONSTANTS ON CARD	

M.P. 42 REV. 6/70

Decimal Wheel = 2

Page No. 1 of 1



olivetti programma 101

Title COS(x)

Date m	Date yr.	Code class	Code	No. of cards	No. of instructions
				1	212

PROGRAM INSTRUCTIONS — CARD NO. 1/1

	REG. 1	REG. 2	REG. F	REG. E	REG. D
1	A V	25	49	73	97
2	S	26	50	74	98
3	↓	27	51	75	99
4	A ↕	28	52	76	100
5	D ÷	29	53	77	101
6	A x	30	54	76	102
7	C ↕	31	55	79	103
8	F / ↓	32	56	80	104
9	C x	33	57	81	105
10	F +	34	58	82	106
11	C x	35	59	83	107
12	E / +	36	60	84	108
13	C x	37	61	85	109
14	E +	38	62	86	110
15	C x	39	63	87	111
16	D / +	40	64	88	112
17	C x	41	65	89	113
18	A / ↑	42	66	90	114
19	D / ↓	43	67	91	115
20	+	44	68	92	116
21	A ◇	45	69	93	117
22	V	46	70	94	118
23		47	71	95	119
24		48	72	96	120
	REG. 1	REG. 2	REG. 11	REG. el	REG. dl

CONTENTS OF REGISTER

M	OPERANDOS
A	
R	
bl	
B	
cl	EN USO
C	EN USO
dl	-4'9347
D	180
el	-1'3323
E	4'0580
fl	-0'0205
F	0'2296

CONSTANTS ON CARD		CONSTANTS ON CARD	
-4'9347	D / ↑	4'0580	E . ↑
180	D . ↑	-0'0205	F / ↑
-1'3323	E / ↑	0'2296	F . ↑

Decimal Wheel = DE 5 A 8

M.P. 42 REV. 6/70

Page No. 1 of 1

olivetti programma 101

Title: ATERRIZAJE LUNAR

Date m	Date yr.	Code class	Code	No. of cards	No. of instructions
				1	81

PROGRAM INSTRUCTIONS — CARD NO. 1/1

	REG. 1	REG. 2	REG. F	REG. E	REG. D
1	A V	25 ↓	49 C ↓	73 A / ↑	97
2	A / ↑	26 A ↕	50 C / ↕	74 R *	98
3	D -	27 D ↕	51 B ↓	75 R *	99
4	E / ↑	28 D ↓	52 B / ↕	76 R *	100
5	A / ↑	29 D / -	53 D ↓	77 R *	101
6	R S	30 / V	54 D / ↕	78 R *	102
7	E -	31 A ↕	55 C V	79 E *	103
8	B / ↑	32 D ↕	56 A / V	80 ◇	104
9	A / ↑	33 E / -	57 A / ↑	81 5	105
10	R S	34 C ↕	58 D ↑	82	106
11	R S	35 C ↓	59 ↓	83	107
12	D -	36 B / +	60 E / X	84	108
13	C / ↑	37 B ↕	61 C / X	85	109
14	A / ↑	38 C ↓	62 D ↕	86	110
15	R S	39 A / ↑	63 B / ↓	87	111
16	R ↑	40 D ↑	64 B / X	88	112
17	D ↓	41 ÷	65 D +	89	113
18	D / ↑	42 B / +	66 A √	90	114
19	B V	43 C / +	67 B / ↕	91	115
20	B / ◇	44 C ↕	68 B W	92	116
21	C / ◇	45 C ↓	69 B / ◇	93	117
22	D / ◇	46 / W	70 A / ↑	94	118
23	/ ◇	47 C W	71 D S	95	119
24	S	48 A / W	72 ◇	96	120
	REG. 1	REG. 2	REG. F	REG. E	REG. D

CONTENTS OF REGISTER

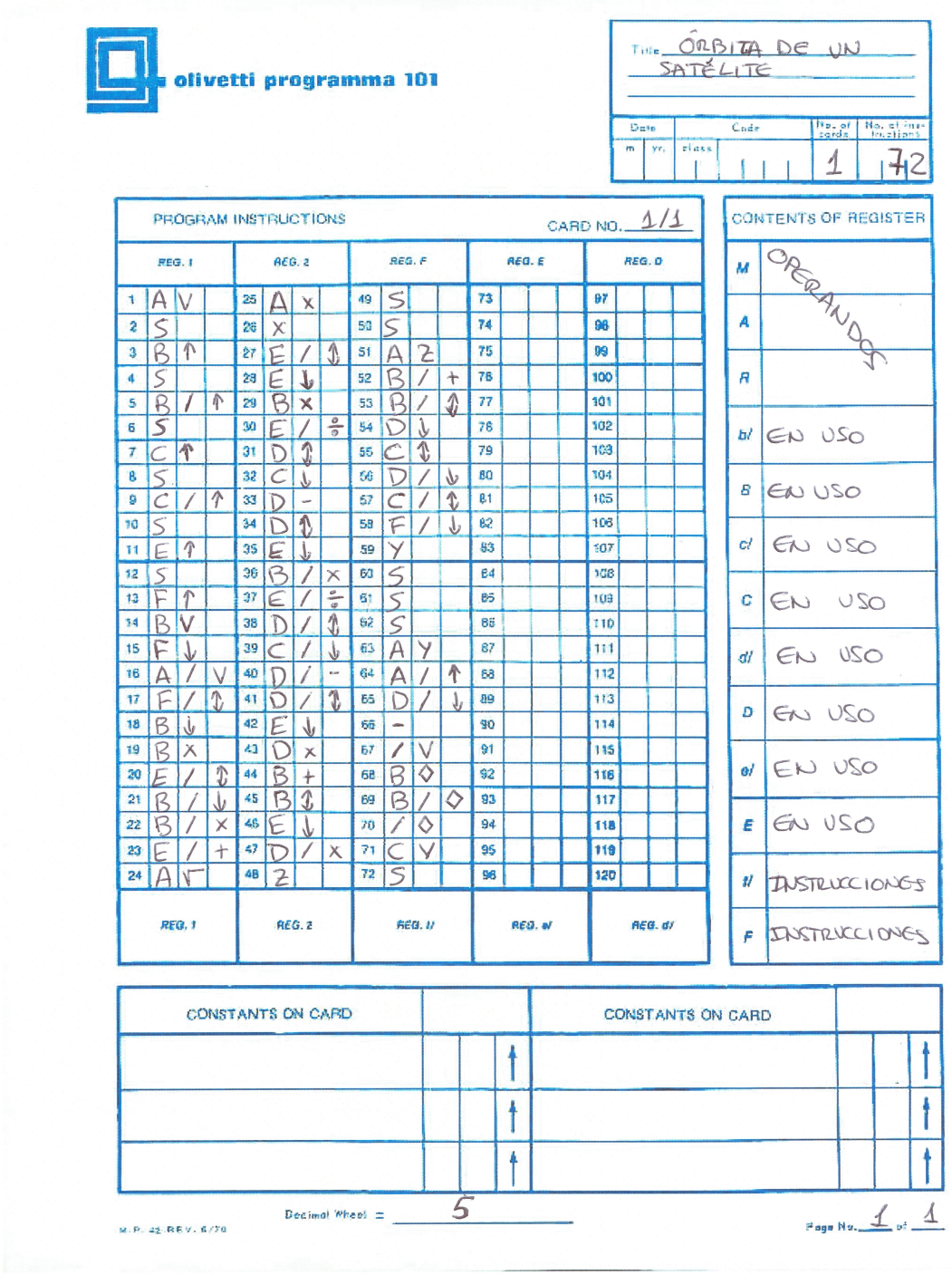
M	OPERANDOS
A	
R	
b/	EN USO
B	EN USO
c/	EN USO
C	EN USO
d/	EN USO
D	EN USO
e/	EN USO
E	INSTRUCCIONES
f/	INSTRUCCIONES
F	INSTRUCCIONES

CONSTANTS ON CARD		CONSTANTS ON CARD	

Decimal Wheel = 2

M.P. 43 REV. 6/70

Page No. 1 of 1



olivetti programma 101

Title: ÓRBITA DE UN SATÉLITE

Date m	Date yr.	Code class	Code	No. of cards	No. of instructions
				1	72

PROGRAM INSTRUCTIONS — CARD NO. 1/1

	REG. 1	REG. 2	REG. F	REG. E	REG. D
1	A V	25 A x	49 S	73	97
2	S	26 x	50 S	74	98
3	B ↑	27 E / ⇕	51 A Z	75	99
4	S	28 E ↓	52 B / +	76	100
5	B / ↑	29 B x	53 B / ⇕	77	101
6	S	30 E / ÷	54 D ↓	78	102
7	C ↑	31 D ⇕	55 C ⇕	79	103
8	S	32 C ↓	56 D / ↓	80	104
9	C / ↑	33 D -	57 C / ⇕	81	105
10	S	34 D ⇕	58 F / ↓	82	106
11	E ↑	35 E ↓	59 Y	83	107
12	S	36 B / x	60 S	84	108
13	F ↑	37 E / ÷	61 S	85	109
14	B V	38 D / ⇕	62 S	86	110
15	F ↓	39 C / ↓	63 A Y	87	111
16	A / V	40 D / -	64 A / ↑	88	112
17	F / ⇕	41 D / ⇕	65 D / ↓	89	113
18	B ↓	42 E ↓	66 -	90	114
19	B x	43 D x	67 / V	91	115
20	E / ⇕	44 B +	68 B ◇	92	116
21	B / ↓	45 B ⇕	69 B / ◇	93	117
22	B / x	46 E ↓	70 / ◇	94	118
23	E / +	47 D / x	71 C V	95	119
24	A Γ	48 Z	72 S	96	120
	REG. 1	REG. 2	REG. I/	REG. e/	REG. d/

CONTENTS OF REGISTER

M	OPERANDOS
A	
R	
b/	EN USO
B	EN USO
c/	EN USO
C	EN USO
d/	EN USO
D	EN USO
e/	EN USO
E	EN USO
i/	INSTRUCCIONES
F	INSTRUCCIONES

CONSTANTS ON CARD		CONSTANTS ON CARD	

M.P. 42 REV. 6/70

Decimal Wheel = 5

Page No. 1 of 1