UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Estudio del diseño y funcionamiento del computador ENIAC

Trabajo Final de Grado

Grado en Ingeniería Informática

Autor: Enrique Osset Vicente

Tutor: Xavier Molero Prieto

3 de julio de 2014

Resumen

El objetivo de este proyecto, inicialmente histórico, es hacer un estudio de la génesis y diseño del ENIAC, computador pionero, en servicio durante diez años hasta mil novecientos cincuenta y cinco, para analizar su contexto, desarrollo, peculiares características y capacidades; el computador electrónico de propósito general ENIAC tomado como punto de arranque, nos pone de manifiesto los avances de la informática al contemplar la situación actual.

Otro objetivo, en cierto modo homenaje al ENIAC, es realizar y probar una batería de programas que muestren de forma práctica sus capacidades en un simulador, para comprender su modo de funcionamiento, de programación, así como lo complejo y limitado que resultaba. Las aplicaciones tienen una marcada componente didáctica.

Palabras clave: ENIAC, primeros computadores, historia de la informática, museo de informática, programación del ENIAC, electrónica.

Índice de imágenes

Índice general

1. Introducción

Este trabajo tiene una parte teórica que describe el origen del ENIAC, con una mirada sobre esos años en la historia de la evolución de los ordenadores, el contexto y las circunstancias que hicieron posible el desarrollo de uno de los primeros computadores electrónicos de propósito general. Se analizan los aspectos técnicos junto con sus capacidades y el avance representado, pues fue el arranque de la evolución de los ordenadores que no ha cesado hasta hoy.

Se considera la coyuntura del momento, la segunda guerra mundial, las necesidades surgidas y el estado de la tecnología electrónica, todo lo cual desembocó en este proyecto llevado a cabo por un equipo heterogéneo de ingenieros universitarios, ver imagen 1, con el laboratorio de balística del ejército americano para fabricar un computador electrónico de propósito general que realizara inicialmente cálculos balísticos, resolviendo las ecuaciones diferenciales de las trayectorias de los cañones para confeccionar las tablas de tiro, pero también una vez concluida la guerra, se podrían abordar otros complejos cálculos que requerían un tiempo prohibitivo a las personas.

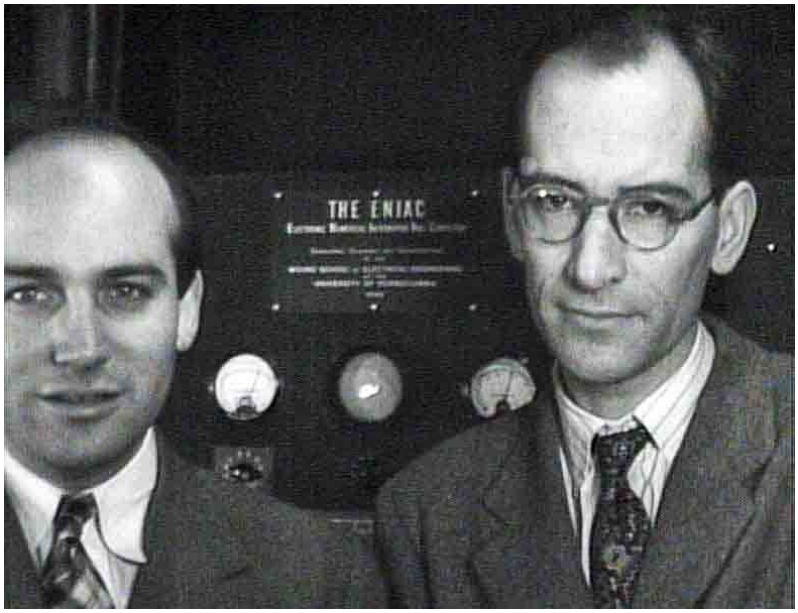


Imagen 1 – Los creadores del ENIAC

Además el proyecto tiene una parte práctica, pues en los estudios previos se ha visto la carencia de programas para el simulador de este ordenador, prácticamente solo hay tres aplicaciones que no acaban de mostrar sus prestaciones, por ello se han desarrollado varios programas que permiten realizar multiplicaciones, divisiones y raíces cuadradas, las capacidades del ENIAC original, a los cuales se han añadido otros como un velocímetro, hallar la

suma de una serie de números, elevar un número al cubo, para concluir con un programa combinado que, a partir de los datos iniciales resuelve el problema físico de una trayectoria, el tiempo de vuelo, la altura máxima y el alcance.

El museo de informática de la UPV (Universidad Politécnica de Valencia) tiene un programa de desarrollo de temas históricos, que contempla ir incluyendo las computadoras más significativas de la historia de la informática, para lo cual los resultados de este estudio sobre el ENIAC se publican en una página multimedia, incluida en el citado museo, al objeto de facilitar su difusión y acceso al público.

2. Motivación y objetivos del trabajo

2.1. Motivación

Este estudio sobre el ENIAC tiene en principio una motivación histórico documental, además de su conocimiento y en cierto modo hacer un homenaje a uno de los primeros ordenadores electrónicos de propósito general; los proyectos que no son actuales, ni innovadores o con poco beneficio, gozan de escasa dedicación. Suelen estar algo abandonados, por ello las personas con afición a estos temas culturales le dedican su esfuerzo para difundirlos.

Por otro lado el ordenador con sus circunstancias resultan simpáticas a los que casi convivieron con él, produciendo cierta nostalgia volver a estudiar aspectos y tecnologías que ya fueron abandonados hace muchos años, como las válvulas de vacío. Además cuando se analiza un equipo tan primitivo, se puede descender a un nivel muy profundo, mientras en un ordenador moderno resulta casi imposible, perdidos entre millones de transistores. Máxime con el ENIAC que se fabricó con los acumuladores y demás componentes sueltos, lo cual obliga a programar hasta las conexiones internas.

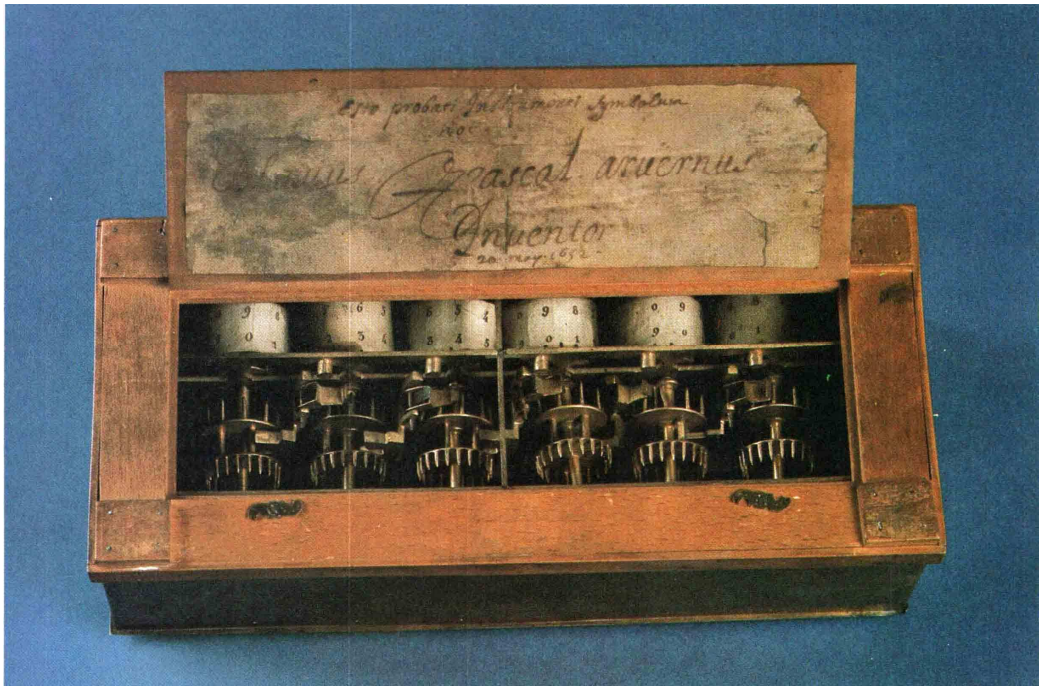


Imagen 2 – La Pascalina

Antes de continuar con el trabajo, conviene situarnos en los años cuarenta del siglo XX y hacer un repaso rápido a la evolución de los computadores hasta ese punto. Partiendo del siglo XVII, se observa un avance de las matemáticas

con el desarrollo de los logaritmos (Ifrah, 2008) que facilitaron los cálculos manuales. Aparecen reglas de cálculo y las máquinas aritméticas como la de Blaise Pascal conocida por la Pascalina, mostrada en la imagen 2; en el siglo XIX Georges Boole desarrolló la lógica simbólica y su álgebra, mientras tanto Charles Babbage trabajó en la máquina analítica, ver imagen 3, antecesor de los ordenadores modernos con una estructura bastante próxima a la arquitectura actual, (Barceló, 2008) que no pudo construirse con la tecnología disponible.

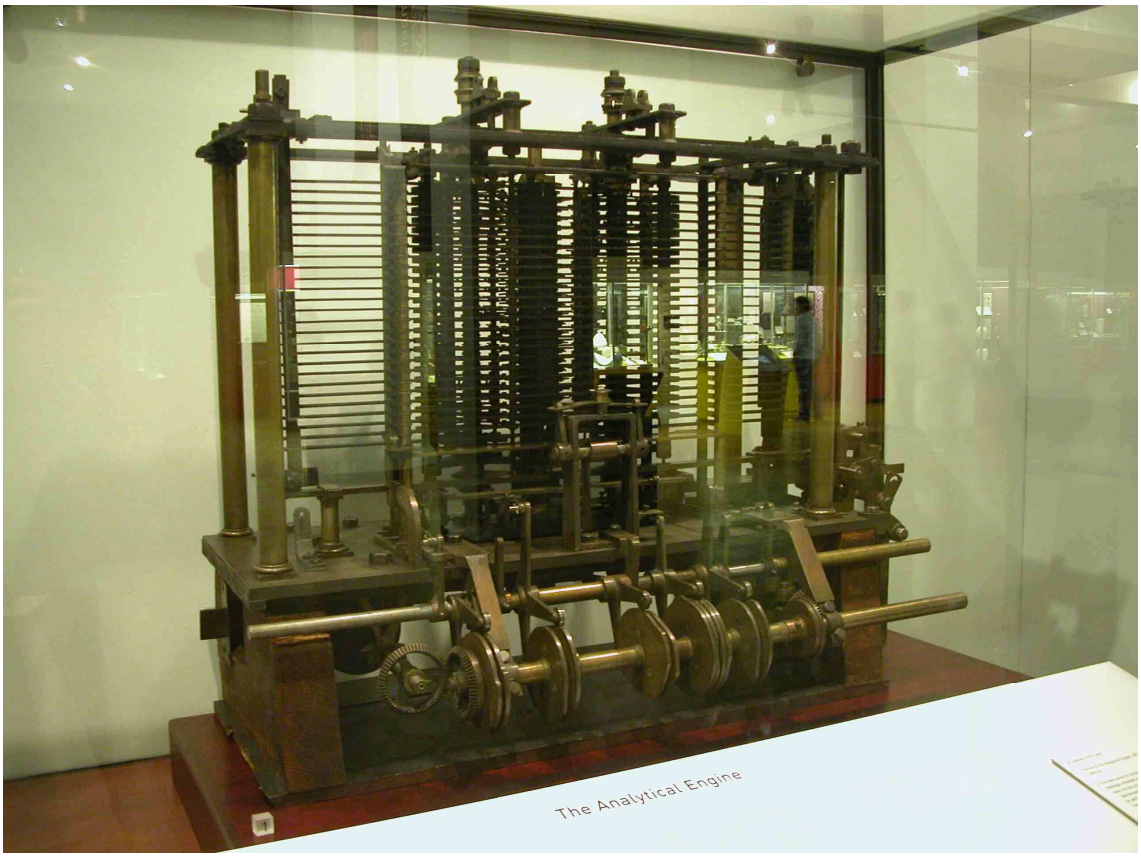


Imagen 3 – La máquina analítica de Babbage

Así llegamos al final del siglo XIX y comienzos del XX, donde la realización del censo norteamericano de 1890 por Herman Hollerith con un equipo compuesto por una lectora de tarjetas perforadas, una clasificadora y una tabuladora para sumar e imprimir los resultados (Breton, 1989), marcó un hito que extendió el uso de tabuladoras en la administración, la industria y el comercio. En paralelo comenzaron a usarse calculadoras analógicas cuya evolución alcanzó su mayor desarrollo en dos modelos, el analizador diferencial de Vannevar Bush (Goldstine, 1993) que se muestra en la imagen 4, para la resolución de ecuaciones mediante un circuito eléctrico, así como la gran calculadora electromecánica Harvard Mark I, imagen 5, de Howard Aiken, construida por IBM entre 1939 y 1944, la cual pesaba cinco toneladas.

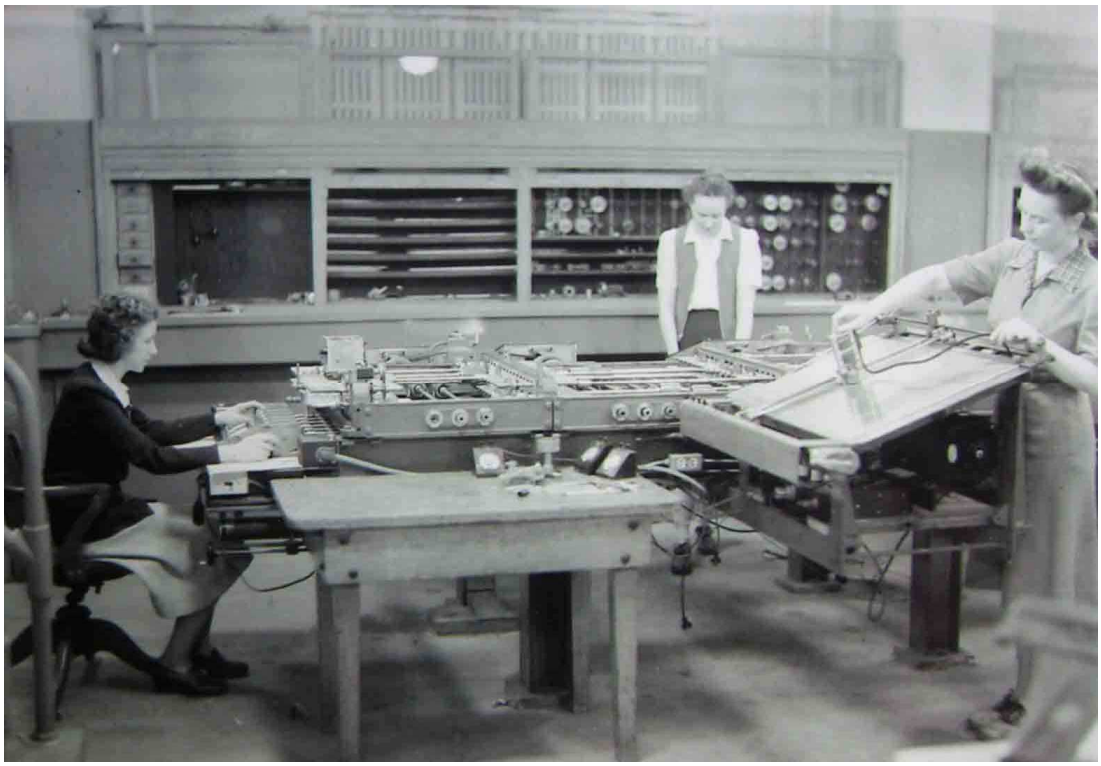


Imagen 4 – K. McNulty, izquierda, con el analizador diferencial

2.2. Objetivos

Los objetivos del proyecto en un principio son investigar y documentar el contexto histórico, el estado de la tecnología y el desarrollo previo de las máquinas calculadoras. Estudiar el diseño y desarrollo del ENIAC con una perspectiva histórica. Buscar la patente del ENIAC, así como las fuentes de información e imágenes o películas del computador, para analizar y aprender su estructura, funcionamiento junto con sus capacidades. Ilustrar gráficamente el texto con detalle usando los planos de la patente, fotografías e imágenes.

A continuación, estudiar y aprender su programación en el simulador, eniac.jar de Till Zoppke, para desarrollar y probar una serie de programas, cubriendo tres niveles. En el primero, alcanzar las capacidades básicas, similares al ENIAC original mediante la implementación de la multiplicación, división y raíz cuadrada. En el segundo nivel realizar un velocímetro, sumar una serie de números y elevar una cantidad al cubo. Y en el tercer nivel, desarrollar un programa integral para calcular una trayectoria a partir de los datos iniciales, usando los elementos disponibles en la versión *Extended large* del simulador.

Todo ello para recoger el trabajo realizado en este documento y desarrollar una página multimedia que se implementará en la página web del museo de informática, explicando los resultados del estudio además de permitir su

difusión y el acceso público, dentro del objetivo general del museo de informática para ir ampliando su información disponible.

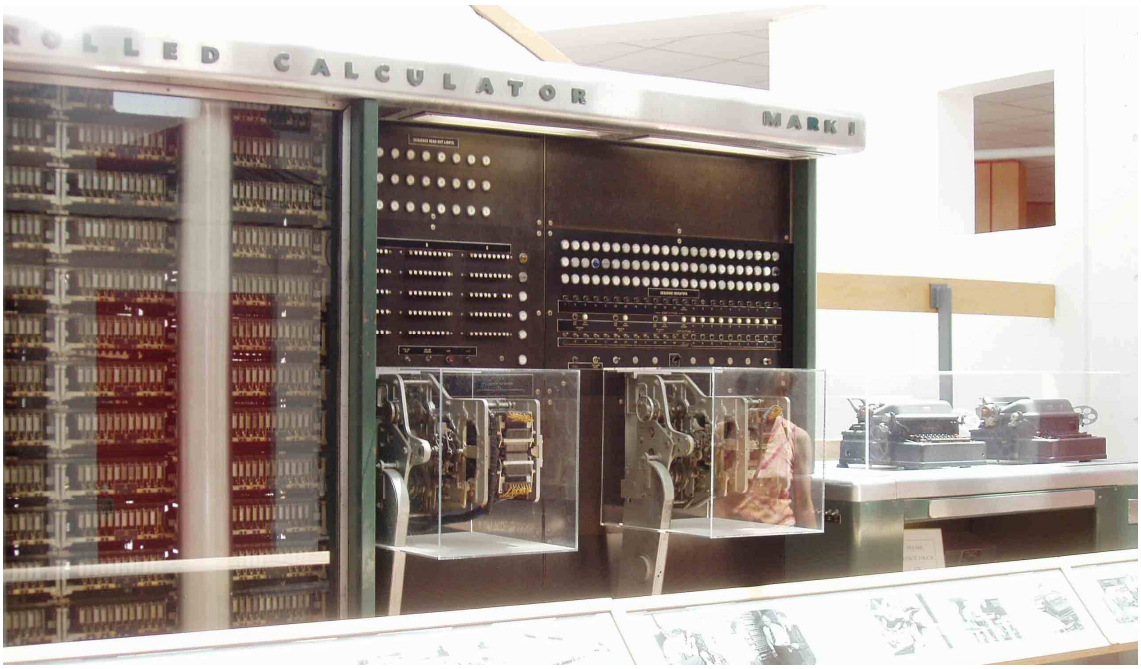


Imagen 5 – El Harvard Mark I de IBM

3. Contexto histórico del ENIAC

El contexto histórico se va desarrollar con unas pinceladas resumidas estructuradas en tres aspectos, uno es el estado de la ciencia y tecnología, electrónica principalmente. Por otro lado se repasa el momento en la evolución de las calculadoras (Swedin y Ferro, 2005), para tratar finalmente el contexto sociológico, pues se construyó en la segunda guerra mundial como resultado del contrato entre el laboratorio de balística con los ingenieros de la *Moore School*.

3.1. Estado de la física y la electrónica

Partiendo del final del siglo XIX se analizan los progresos de la física y la electrónica. Thomas Alva Edison mejorando las lámparas de incandescencia, o bombillas, dicho de forma sencilla introdujo una placa que hacía de ánodo cerca del filamento o cátodo, produciendo al calentarse una corriente; así descubrió el efecto Edison o termoiónico, pues esa "bombilla" se comportaba como un diodo entre el filamento y la placa, aunque Edison no era consciente del alcance.

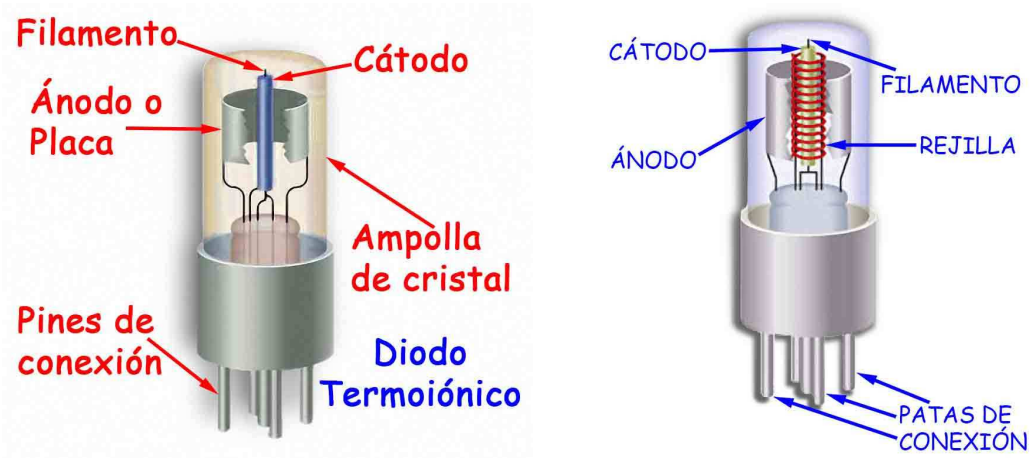


Imagen 6 – Válvulas termoiónicas, diodo y triodo

Esto permitió a John Ambrose Fleming de la Marconi Company desarrollar en 1904 la válvula de vacío o termoiónica, que era un diodo de vacío y básicamente rectificaba la corriente. Luego Lee de Forest en 1906 intentando amplificar la corriente de la válvula, introdujo una rejilla de platino entre el ánodo y cátodo la cual, polarizándola adecuadamente le permitía amplificar la corriente entre ambos, así inventó el tríodo. Ambos diodo y tríodo se muestran en la imagen 6. A partir de estos, se desarrollaron gran variedad de tipos, tetrodos, pentodos, etc, sobre los cuales no se va a profundizar. Además desde

1918 se disponía del biestable (Augarten, 1984) o *flip flop* inventado por Williams Eccles y F.W. Jordan que constaba de dos válvulas.

En la primera mitad del siglo XX la física y la electrónica se apoyaron en la tecnología de circuitos con válvulas de vacío que constituía el mayor avance pero presentaba serias carencias como el excesivo tamaño, tenían una vida limitada pues el filamento se fundía o evaporaba, generaban mucho calor y necesitaban un elevado voltaje que agravaba el problema del calor disipado. Con ellas se construyeron las radios de válvulas que prácticamente eran con la luz eléctrica, los elementos disponibles en los hogares. En la industria había algunos aparatos eléctricos y pocas calculadoras. Las universidades estaban desarrollando inventos, aparatos de mediciones y contadores de pulsos, existiendo el rádar como uno de los avances punteros en tecnología, pero no estaban extendidos.

Por otro lado la telefonía comenzó su andadura desde 1857 de la mano de Antonio Meucci, estando bastante implantada en la sociedad, con otros circuitos propios para los aparatos y las centralitas telefónicas. Utilizaban una tecnología electromecánica menos avanzada que las válvulas, pero eran bastante robustas, más económicas y tenían menos averías si bien más lentas. Era la segunda tecnología disponible. En sus principios fue usada para fabricar calculadoras por inventores sin apoyo financiero que construían sus prototipos como George Stibitz, físico de la Bell Telephone Laboratories, ver imagen 7, el cual fabricó en 1939 el *complex calculator* o *BTL Model 1*, (Shurkin, 1996) aunque otros como Zuse la tuvo que usar en el Z3 y Z4, pues en Alemania hasta después de la guerra fue difícil conseguir válvulas, incluso el ENIAC tenía mil quinientos relés.

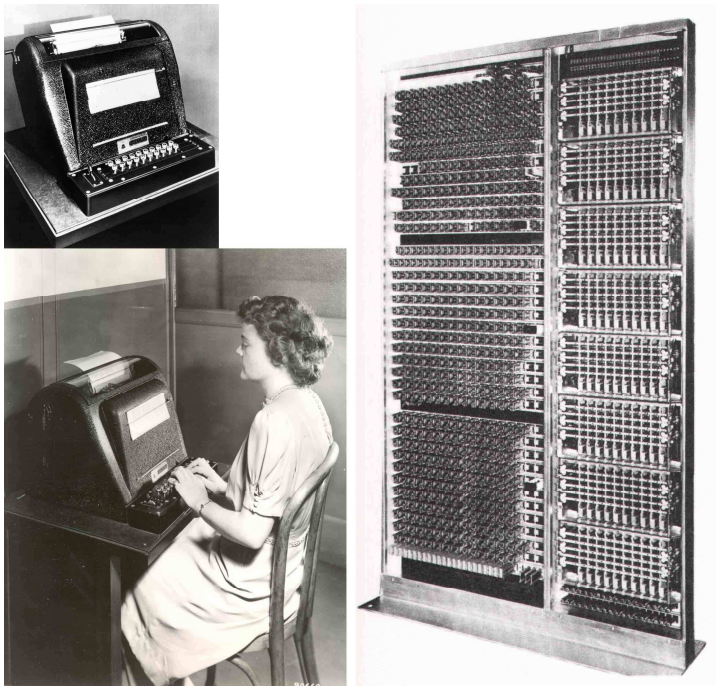


Imagen 7 – El *complex calculator* o *BTL Model 1*

3.2. Desarrollo de las calculadoras

Se hace un paréntesis para reflexionar y situarnos en los años cuarenta del siglo XX. No había ordenadores ni informáticos, casi todo se calculaba a mano, con tablas de logaritmos y algunos tenían acceso a calculadoras manuales o eléctricas, entre lo más avanzado estaba el analizador diferencial del cual solo se construyeron siete. Mientras tanto la ciencia y la ingeniería seguían progresando, ello requería hacer cálculos cada vez más complejos. Esta necesidad empujó la evolución de las calculadoras, en principio mecánicas, que fueron mejorándose con la electrónica y otras, hasta llegar a los computadores.

Imagen 8 – Réplica del Z3 (1941) y Z4 (1944) de Konrad Zuse

La situación se puede resumir en que había unos científicos matemáticos, físicos e ingenieros, sobre todo los aeronáuticos, que cada vez más requerían calculadoras de mayor potencia. En esos años, se solía denominar calculador a la persona que los realizaba, la cual necesitaba sólidos conocimientos matemáticos, físicos o de ingeniería para llevarlos a cabo, hoy el término calculadora o *computer* se suele referir al aparato. Así el diseño de las máquinas acabó en manos de los electrónicos, tanto físicos como ingenieros, pues tuvieron que construir las calculadoras (Ceruzzi, 2003) según se les añadía electrónica.

Esta dinámica produjo una evolución desde la calculadora al computador, donde resulta difícil clasificar los aparatos intermedios o decidir quién fue el primero, por ello se enumeran los hechos evitando el debate. Se comienza con Konrad Zuse Ingeniero por la Universidad Técnica de Berlín, que en 1938 construyó en su casa el Z1, una calculadora binaria mecánica y eléctrica, programable con una cinta perforada, no funcionaba correctamente debido a las piezas mecánicas. Desarrolló el Z2 (Wiliams, 1997) similar al Z1, que corrigió el problema con tecnología de relés telefónicos.

El año 1941 fue movilizado Zuse en la fábrica de aviación Henschel, donde construyó el Z3 de una tonelada que usaba relés, es el primer computador electrónico de propósito general con programa que usaba aritmética binaria de coma flotante, tenía una memoria de 64 palabras de 22 bits y el programa se introducía mediante un lector de bandas. Destruido en 1943 durante un bombardeo de Berlín. Construyó en 1944 el Z4 de una tonelada que leía tarjetas perforadas, era programable, con una memoria de 500 palabras de 32 bits, tenía entrada de datos por teclado o tarjetas perforadas y salida por una máquina de escribir, en la imagen 8 se muestran ambos computadores con su creador.

Imagen 9 – Réplica del Atanasoff Berry *Computer, ABC*

El año 1939 Atanasoff y Berry diseñaron en la Universidad de Iowa el ABC, Atanasoff Berry *Computer*, ver imagen 9, de unos trescientos veinte kilos, para resolver ecuaciones lineales que no era programable, usaba el sistema binario, era electrónico pero tenía un reloj interno de sólo sesenta pulsaciones por segundo, el ABC no funcionaba correctamente, tenía un error cada cien mil cálculos. Por otro lado, en septiembre de 1940 Stibitz realizó una demostración de cálculo a distancia, para ello conectó en Hanover por teletipo con la *BTL Model 1* en Nueva York, asistiendo von Neumann y Mauchly entre otros.

A partir de 1944 los ingleses, ver imagen 10, dirigidos por Tommy Flowers instalaron en Bletchley Park la serie Colossus, fueron diez calculadoras electrónicas para descifrar los mensajes de los alemanes, tenían una programación limitada pero diseñadas para el criptoanálisis, no eran de propósito general; se basaban en la definición teórica de la máquina de Alan Turing, matemático que también trabajó en el Bletchley Park. Las Colossus eran secreto militar, se conocen poco y fueron destruidas al acabar la guerra.

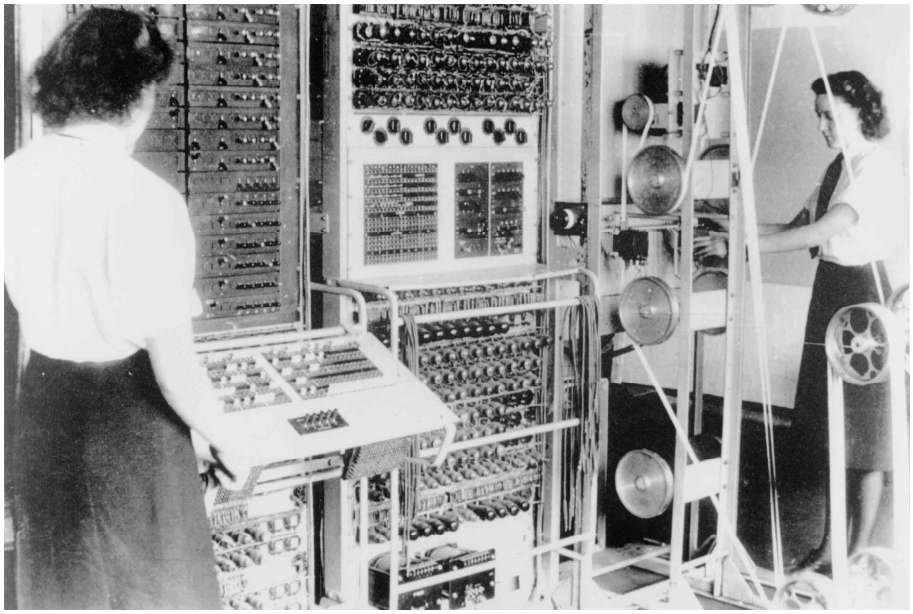


Imagen 10 – El Colossus Mark II

3.3. Contexto sociológico

La segunda guerra mundial comenzó en 1939, como consecuencia de la misma se desarrollaron en Estados Unidos muchos proyectos para alcanzar la supremacía, entre otros el desarrollo de armas nucleares conocido como proyecto Manhattan y en otro orden, los computadores para conseguir modelos electrónicos de propósito general, más rápidos y capaces de hacer cálculos de

todo tipo (Campbell-Kelly, 2004), así como emplearlos para avanzar en inteligencia artificial.

Al margen de esto, John W. Mauchly ingeniero de la *Moore School of Electrical Engineering* el verano de 1941 visitó a Atanasoff examinando su calculadora ABC. En 1942 publicó el memorándum "*Utilización de tubos al vacío de alta velocidad para realizar cálculos*". El laboratorio de balística del ejército, *BRL*, se interesó por la idea y crearon el proyecto secreto "PX" en 1943 para desarrollar el ENIAC, *Electronic Numerical Integrator and Computer*, entre la *Moore School* de Filadelfia, con J. Presper Eckert, ingeniero jefe y John W. Mauchly ingeniero consultor, que aparecen en la imagen 11, con John Brained supervisor administrativo y Herman H. Goldstine matemático y capitán representante del laboratorio.

Imagen 11 – J. Presper Eckert y John W. Mauchly

El proyecto fue contratado según figura, para resolver de manera rápida las ecuaciones de las trayectorias de los cañones o dicho de otra forma, para realizar los cálculos de las tablas de tiro artilleras, pero su primer trabajo en 1945 fueron unos cálculos secretos de los físicos Stanley P. Frankel y Nicholas Metrópolis para Edward Teller sobre la viabilidad de la bomba H, de hidrógeno. El ENIAC se suele publicar que costó 486.000 $, unos 6.000.000 $ actuales, aunque con el índice de precios de consumo, IPC, español hasta 1954 y proyectándolo a 1944, los 486.000 $ serían 54.000.000 $ actuales, que son unos 39.500.000 €.

El ENIAC se comenzó en julio de 1943 y se terminó en noviembre de 1945, fue construido por un equipo heterogéneo, por un lado estaban los ingenieros de la *Moore* financiados por el *BRL* que mantenían intereses comerciales para patentar su trabajo. Estos diseñaron el computador electrónico más avanzado y mayor realizado hasta ese momento, era único. Conocían la complicada maquina, ellos experimentaron analizando las opciones y los problemas que

resolvieron creativa e ingeniosamente con criterios de pragmatismo y con una gran premura de tiempo que les hizo congelar el diseño, lo cual impidió adoptar las mejoras que descubrieron, como la memoria de líneas de retardo, *Delay line memory*, inventada por Eckert en 1944, o corregir los defectos surgidos.

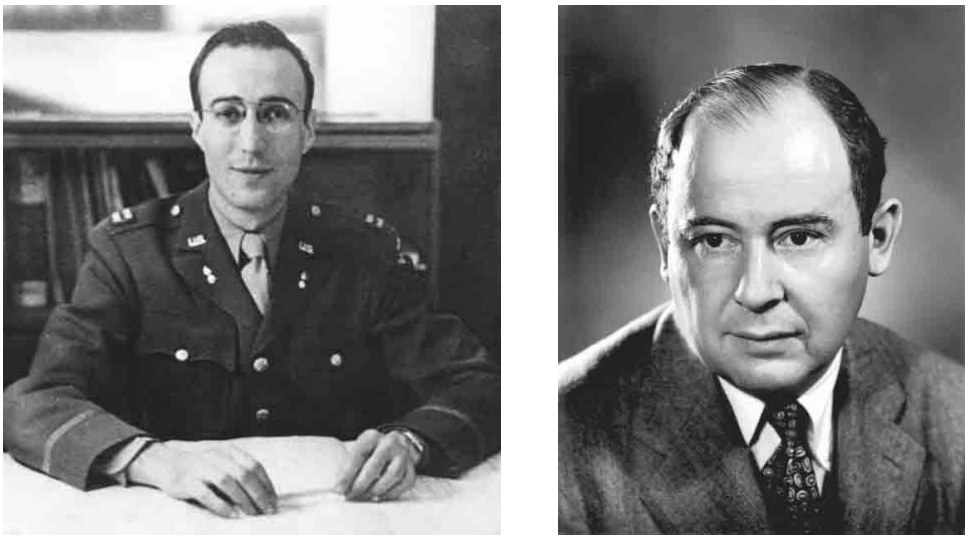


Imagen 12 – Herman H. Goldstine y John von Neumann

Por otro lado estaba el laboratorio de balística, *BRL*, con su representante Herman H. Goldstine matemático que participaba en el proyecto. Estaba casado con Adele Kant matemática que se encargaba de la formación de las programadoras y escribió el manual de operadora, *A Report on the ENIAC*. El computador, bastante complicado de programar, era manejado por seis operadoras seleccionadas entre las calculadoras del *BRL*.

Goldstine, ver imagen 12, en verano de 1944 se reunió con John von Neumann matemático destacado y científico poderoso, explicándole el computador ENIAC al cual le interesó, pues estaba trabajando en el proyecto Manhattan con la bomba atómica que le requería hacer complejos cálculos, por ello desde septiembre se reunió varias veces con el grupo. Neumann también era experto en explosivos, entre otros, diseñó el método de implosión usado en la ciudad de Nagasaki, así mismo calculó la altura a la que debían explotar las bombas para que su efecto fuera más devastador.

Eckert y Mauchly con su equipo desarrollaron el ENIAC, analizaban los problemas, las soluciones adoptadas y las mejoras posibles; parece ser que ya se dieron cuenta de la necesidad del programa almacenado, la escasez de memoria y otros aspectos (McCartney, 1999), que recogieron para desarrollar el siguiente ordenador EDVAC, *Electronic Discrete Variable Automatic Computer*.

Neumann acudía a la *Moore* como consultor, pudo examinar directamente la evolución del computador, tenía reuniones con los ingenieros y redactó el

manuscrito de *First Draft of a Report on the EDVAC* como una memoria del grupo de trabajo para diseñar el futuro EDVAC. El informe fue remitido a Goldstine que lo mecanografió quitando toda referencia a Eckert y Mauchly, recoge lo que se conoce como principios de la arquitectura von Neumann para computadores, difundido en 1945, con una lealtad y objetividad que no se trata.

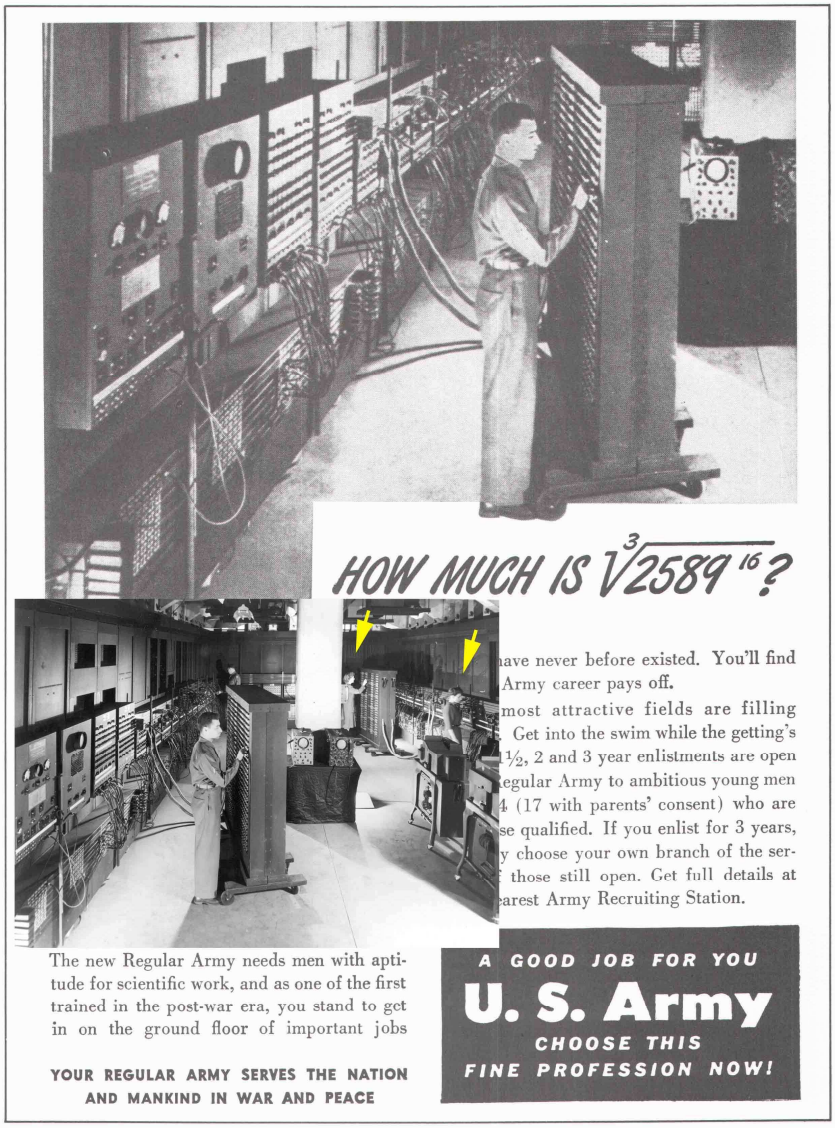


Imagen 13 – Anuncio del ENIAC y la fotografía completa

En febrero de 1946 el ENIAC que ya se empleaba desde agosto de 1945 para los cálculos de la bomba H, fue presentado al público en la *Moore School* de Filadelfia, la prensa (War Department, 1946) elogiaba el computador electrónico y se ensalzaban sus cualidades como cerebro electrónico. El ejército hacía anuncios para reclutar hombres que trabajaran en el ENIAC, algunos con un carácter propagandístico, como el mostrado en la imagen 13 donde se aprecian las programadoras "invisibles" desaparecidas de la foto original y era

algo exagerado al preguntar cuánto es la raíz cúbica de 2589 elevado a la dieciséis (que tiene cincuenta y cuatro dígitos), pues resultaba difícil calcularlo con este computador que solo hacía raíces cuadradas y manejaba números de hasta veinte dígitos.

Eckert y Mauchly continuaron con la construcción del EDVAC pero en marzo de 1946 se fueron de la *Moore* por discrepancias con las nuevas normas implantadas sobre patentes, montando su propia compañía y fabricando ordenadores comerciales como el *BINAC*. El ENIAC fue trasladado en 1947 al polígono de pruebas de Aberdeen, siendo modificado en 1948 como computador de programa almacenado y siguió funcionando hasta el 2 de octubre de 1955.

En estos primeros tiempos previos a la informática, se aprecia una separación entre la teoría y la práctica. La teoría se planteaba por matemáticos (Coello, 2003) como Turing en 1936 con su máquina Turing o von Neumann en 1945 con los principios de la arquitectura de los ordenadores, donde señalaban de manera general la lógica o la estructura de los modelos teóricos (Hally, 2005) a desarrollar. Pero luego en la práctica, eran los físicos o ingenieros electrónicos quienes materializaban esas ideas, incluso algunas de ellas como la inteligencia artificial que ya apuntaron Turing y von Neumann, no resultó factible.

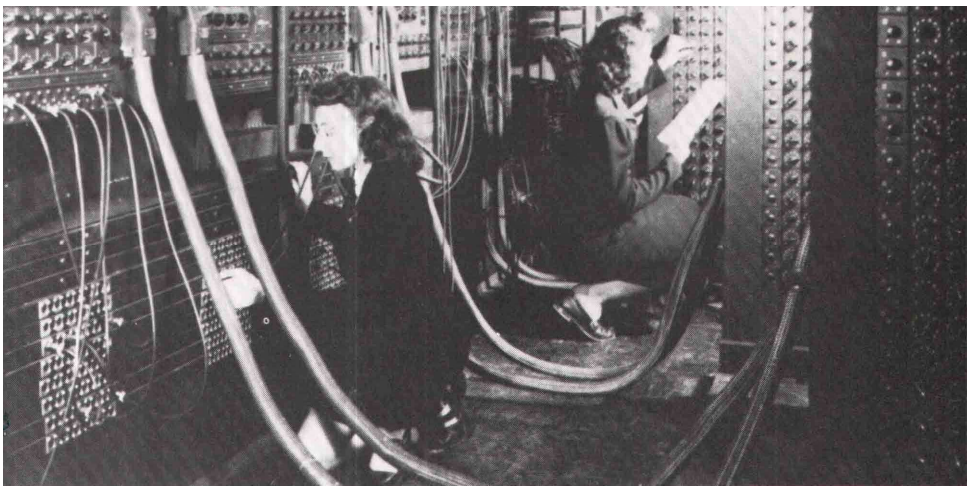


Imagen 14 – Las programadoras cableando el ENIAC

En el apartado de personas olvidadas se menciona que en esos años consideraban a las mujeres más hábiles calculando y había muchas en el laboratorio de balística, por ello al fabricar el ENIAC seleccionaron como programadoras (Light, 1999) a las calculadoras Betty Snyder, Jean Jennings Bartik, Kathleen McNulty, Marilyn Wescoff, Ruth Lichterman y Frances Bilas. La programación era muy complicada por las limitaciones y peculiaridades del ENIAC, como puede verse en la imagen 14, aunque ellas consiguieron hacerla accesible y menos compleja, creando las primeras rutinas y aplicaciones, pero fueron ignoradas hasta los años ochenta.

4. Aspectos tecnológicos del ENIAC

4.1. Generalidades

El ENIAC era un computador muy grande y pesado comparado con sus predecesores, los Z3 y Z4 de Zuse o el ABC de Atanasoff. El ENIAC fue un proyecto secreto, por lo que algunos aspectos no se conocen a pesar de la abundante bibliografía. Su elevado coste y tamaño hacen dudar de su finalidad solo para calcular tablas de tiro, máxime cuando fue usado por los físicos nucleares que tenían una gran necesidad de capacidad de cálculo. Electrónico de propósito general no tenía inicialmente programa almacenado, parece un banco de pruebas para desarrollar las computadoras y es el germen del EDVAC que ya se ha considerado un ordenador con toda su arquitectura.

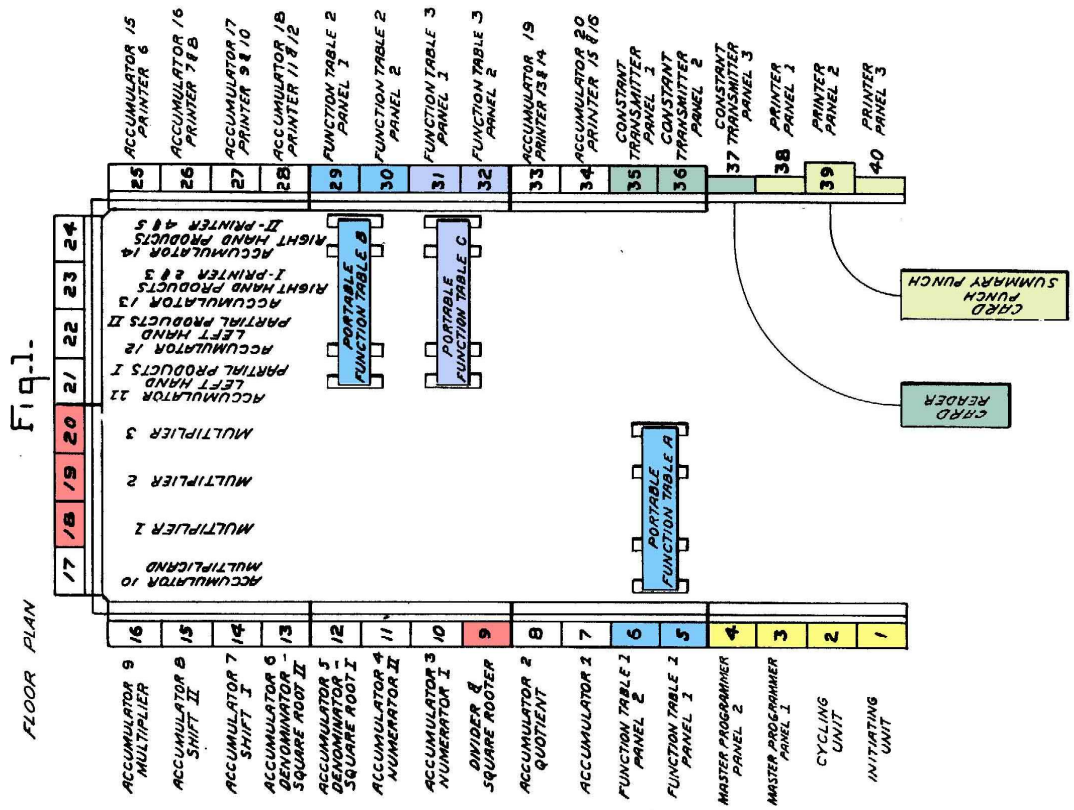


Imagen 15 –Distribución en planta del ENIAC

De la patente del ENIAC (Eckert y Mauchly, 1964) se extraen los planos en blanco y negro, por facilidad se mantiene la numeración de la figura original con sus explicaciones. La imagen 15 con la figura 1 el esquema de planta, muestra la estructura del ENIAC con cuarenta componentes de 0,6 metros de ancho por 2,7 de altura y 0,7 de profundidad, dispuestos en forma de U, ocupaba una habitación de unos 10 x 17 metros y pesaba sobre 27 toneladas. Además sin

numerar aparecen las tres tablas de función portátiles, la lectora y la perforadora de tarjetas de IBM, como se ve en la vista general de la imagen 16.

El ENIAC estaba construido con 17.468 válvulas de vacío, 7.200 diodos de cristal y 1.500 relés, usaba 44 tipos diferentes de cables de conexión, funcionaba con corriente alterna de 240 V trifásica, consumía una potencia de 174 kW máxima y necesitaba refrigeración. Usaba una aritmética decimal de punto fijo, era muy flexible y podía funcionar en paralelo, almacenaba 20 palabras de diez dígitos en los acumuladores y 312 en las tablas de función, además para la entrada y salida usaba tarjetas perforadas de 80 columnas, donde cada una permitía 8 números de 10 dígitos. Para hacerse una idea de la envergadura cada componente, de media pesaba 650 kilos, tenía 436 válvulas y consumía 4 kW.

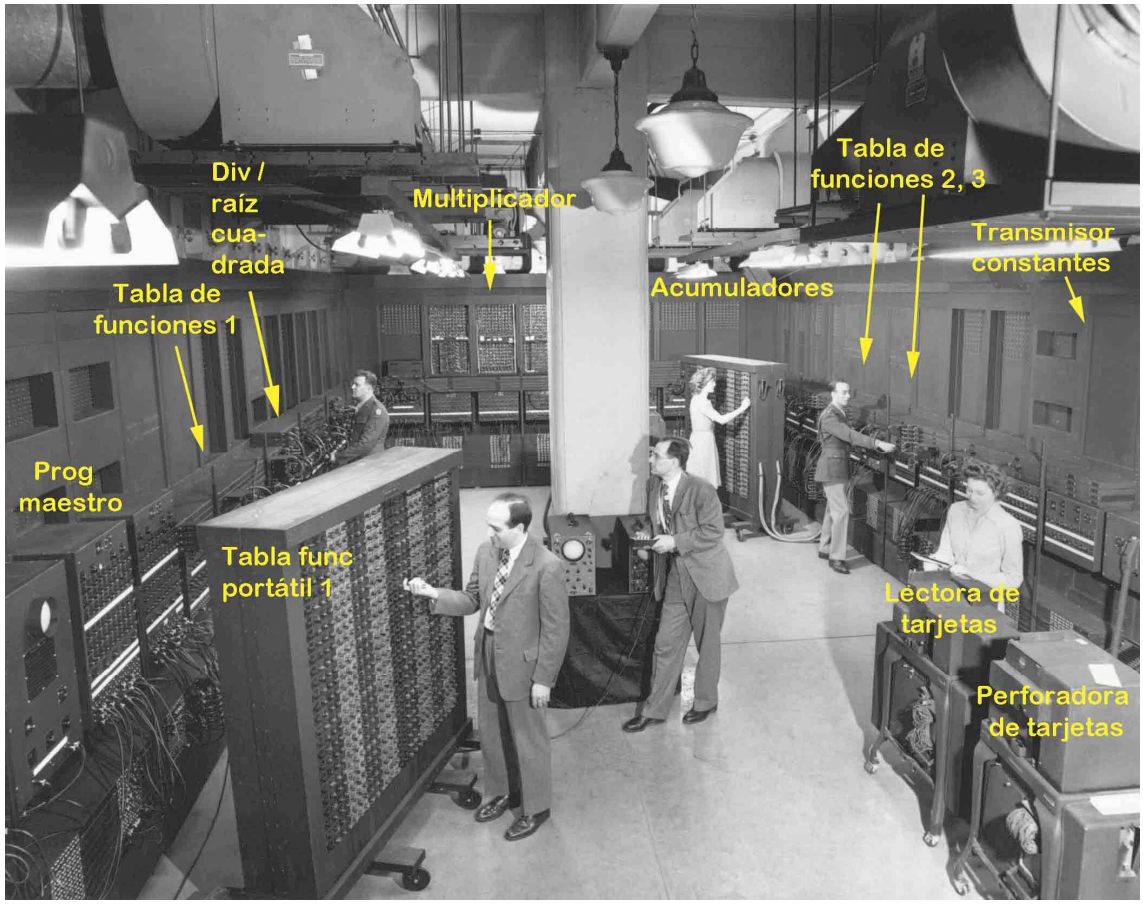


Imagen 16 – Vista general del ENIAC

Sus prestaciones de cálculo (Brainerd y Sharpless, 1999) eran de una suma cada 200 microsegundos, unas cinco mil por segundo. Una multiplicación cada 2.800 y una división o raíz cuadrada cada 24.000 microsegundos aproximadamente, pues dependía del número de factores y la precisión requerida. Podía leer 125 tarjetas perforadas por minuto o escribir 100. Además,

para trabajar con números de 20 dígitos, se podían unir dos acumuladores. El tiempo medio sin fallos era de 5 horas y 36 minutos.

Su arquitectura como se ve en la imagen 15, se suele dividir en cinco partes (Molero, 2013). La de *control global y programación* (en amarillo) con la unidad de inicio, la unidad de ciclos y el programador maestro. La de *aritmética y almacenamiento* (en blanco y rojo) con veinte acumuladores, el multiplicador de alta velocidad y el divisor/raíz cuadrada. La *memoria* (en azul) con tres tablas de función. La *entrada/salida* (en verde) con el transmisor de constantes, la lectora de tarjetas, tres paneles de impresora y la perforadora de tarjetas. Y los *buses* con el bus de programa, el de datos y el bus de sincronización.

4.2. Válvulas de vacío

La tecnología del ENIAC se basaba en las válvulas de vacío que usaba de muchos tipos, los acumuladores tenían biestables con válvulas 6NS7, las funciones lógicas usaban 6L7, 6SJ7, 6SA7 y 6AC7 y para conducir impulsos por cables la 6L6 y 6V6. Estas válvulas tenían una duración limitada, se fundían o evaporaban con el uso, por ello se empleaban por debajo de sus prestaciones para alargar la vida, pues hasta 1948 no aparecieron válvulas de alta fiabilidad.

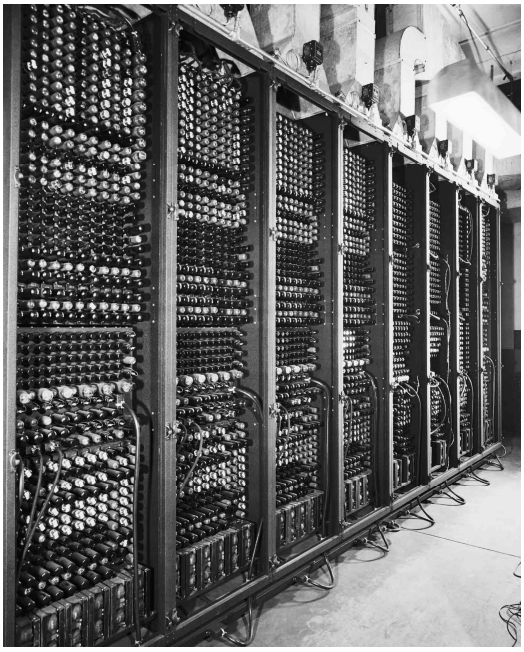


Imagen 17 – Vista de la parte trasera del ENIAC

Cuando se fundía una válvula el problema era menor pero sus consecuencias graves pues se paralizaba el ordenador, si estaba haciendo un cálculo había que repetirlo y costaba mucho tiempo dar con la válvula fundida.

Por ello, para mejorar la fiabilidad y disponibilidad del computador, una preocupación constante desde el principio, llevaba unos relojes del tiempo de funcionamiento y tenía un diseño modular de tal manera que al fallar una parte, se cambiaba por detrás directamente el módulo, como se ve en la imagen 17, así en unos quince minutos el ordenador se reparaba y podía seguir funcionando.

4.3. El control global y la programación

El control global y la programación incluían la unidad de inicio, de ciclos y el programador maestro, como se ven en la imagen 18 y 22. La unidad de inicio y la de ciclos dirigían el funcionamiento del resto de componentes del ENIAC. En la imagen 20, figuras 24 y 28 de la patente, se destacan en amarillo las partes de dichas unidades implementadas por la simulación eniac.jar de Till Zoppke.

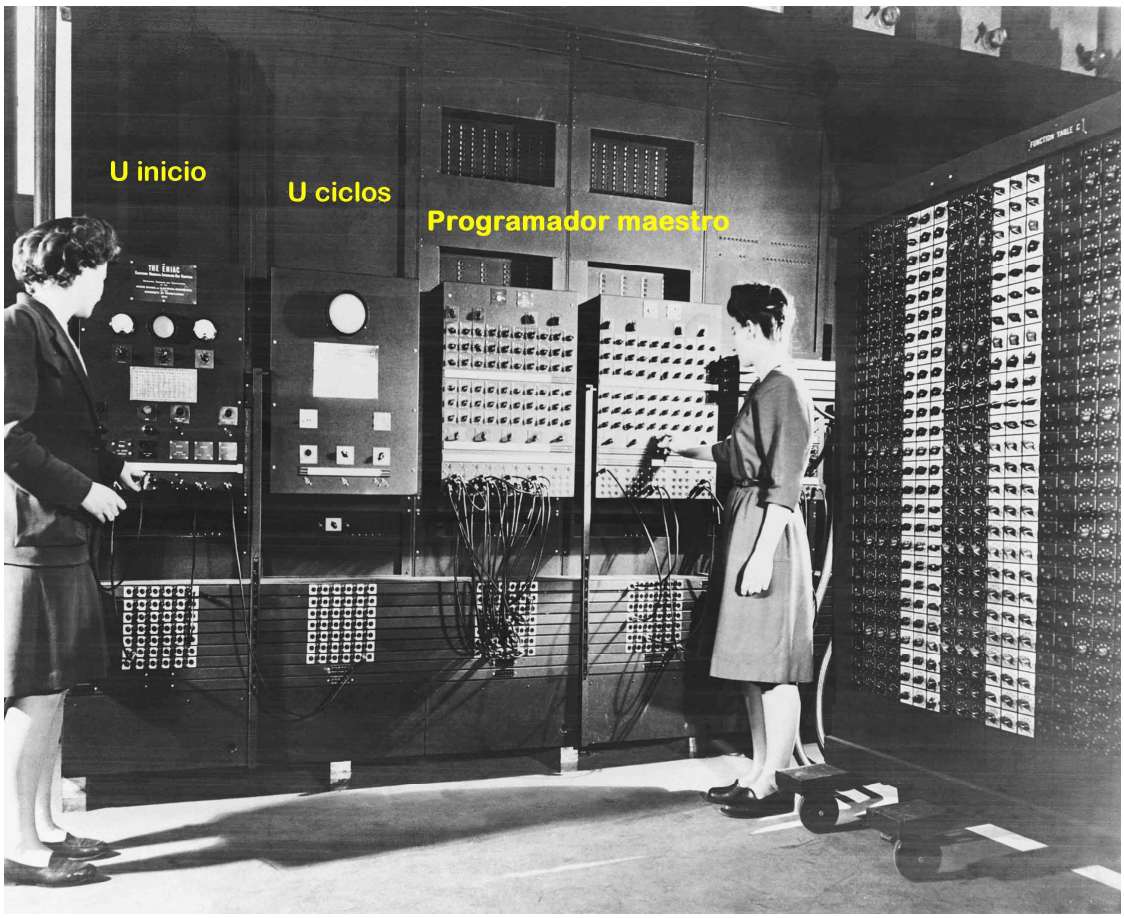


Imagen 18 – El control global y la programación del ENIAC

La unidad de inicio mostrada en las imágenes 19 y 20 con las figuras 23 y 24 de la patente, tenía pulsadores manuales para el arranque y parada general del ENIAC, el pulso inicial de programa, (Martin, 1995) el borrado, tanto inicial como un borrado selectivo de los acumuladores activados y otra serie de

pulsadores. En la parte superior estaban los interruptores para seleccionar las corrientes continuas y alternas que usaba de varios tipos, pues tenía sus fuentes de alimentación accesorias que no se van a tratar.

La unidad de ciclos que aparece en las imágenes 19 y 20 era el reloj del ENIAC, proporcionaba los pulsos que coordinaban todos los elementos a través del bus de sincronización, en lo que se conocía como un ciclo de suma (Rojas y Hashagen, 2000) que duraba doscientos microsegundos dividido en veinte segmentos. Era complicado pues tenía diez trenes de pulsos diferentes con cometidos distintos cada uno, estaban formados casi todos ellos por pulsos individuales de dos microsegundos a cincuenta y cinco voltios.

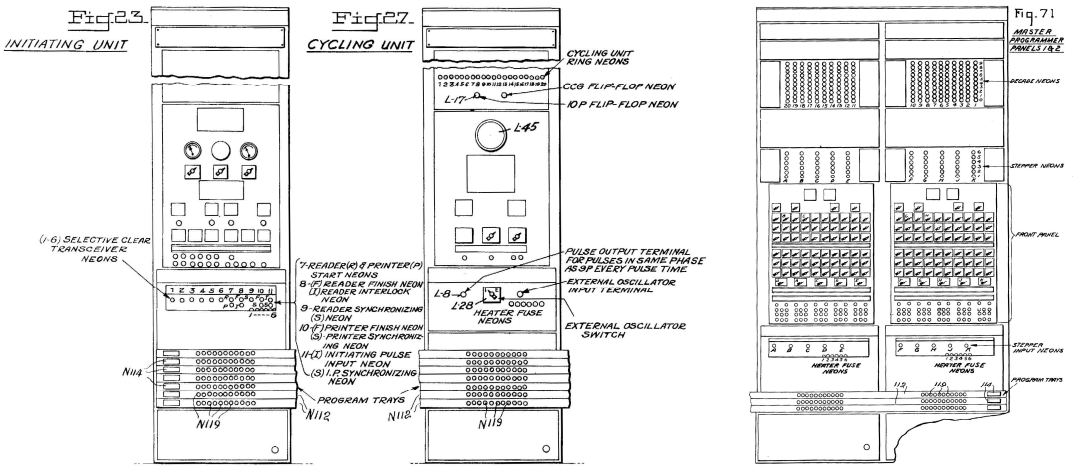


Imagen 19 – Las unidades del control global y programación

La unidad de ciclos tenía un interruptor de encendido, *Heaters*, un selector de operación que podía funcionar de forma continua o bien por pasos, con dos opciones de un pulso o de un ciclo de suma, esto le permitía depurar errores. Se lanzaban con un botón pulsador. También había un interruptor del osciloscopio.

El principal tren de pulsos era el *Central Program Pulse CPP,* de 5 kHz, marcaba el arranque de cada ciclo de suma, pero en el microsegundo ciento setenta, como se puede apreciar en la imagen 21. Otro tren importante era el *Reset Pulse, RP* con dos pulsos, cuando un contador de décadas contaba más de diez marcaba un acarreo y en su segundo pulso, microsegundo ciento noventa y posterior al pulso *CPP,* hacía efectivo el acarreo del ciclo anterior. También se hacía efectivo el contador de signo P/M; la imagen 21 lo lleva destacado en azul.

Los trenes de pulsos 10-P y 9-P se usaban por los acumuladores para transmitir los números, los trenes 1-P, 2-P, 2´P y 4-P se empleaban por el transmisor de constantes para inicializar los acumuladores y por las tablas de multiplicación y el tren 1´P era el uno necesario para pasar de complemento a

nueve a complemento a diez. El tren de pulsos *Carry Clear Gate, CCG* en realidad es una puerta de 70 microsegundos, permitía el acarreo y se utilizaba para limpiar los acumuladores con el interruptor de limpieza correcta puesto.

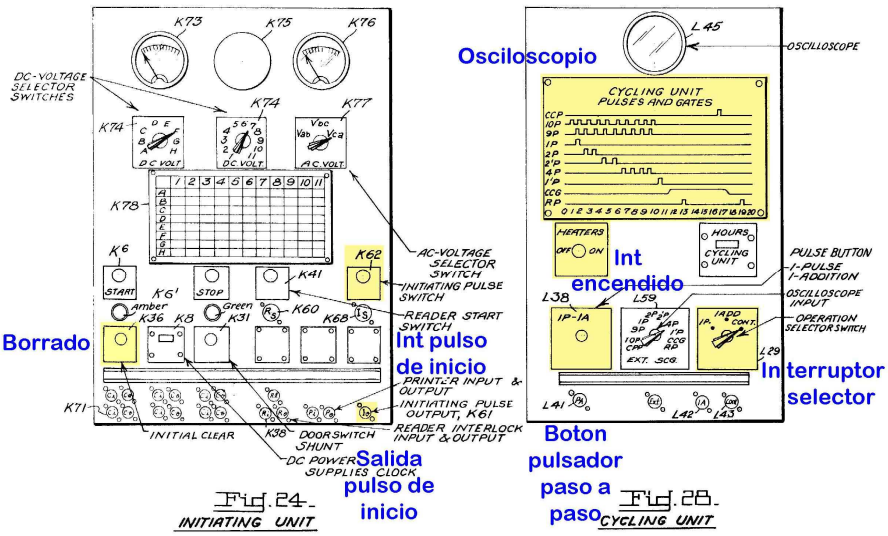


Imagen 20 –Detalle de la unidad de inicio y la de ciclos

El tiempo de duración de las instrucciones del ENIAC era variable. Se tomaba como unidad de medida un ciclo de suma, imagen 21, de doscientos microsegundos, pero dependiendo de las tareas en que se descomponían, usaban los ciclos de suma requeridos. La suma y la resta necesitaban un ciclo. La multiplicación de un número de diez dígitos por un factor de d dígitos, usaba d más cuatro ciclos. La división y la raíz cuadrada con un cociente o raíz de d dígitos, necesitaban d más uno por trece ciclos de media. La memoria leía o escribía en un acumulador con un ciclo, pero para leer en una tabla de funciones v veces, empleaba v más cuatro ciclos. Además el programador maestro y los acumuladores se podían configurar para repetir secuencias de acciones.

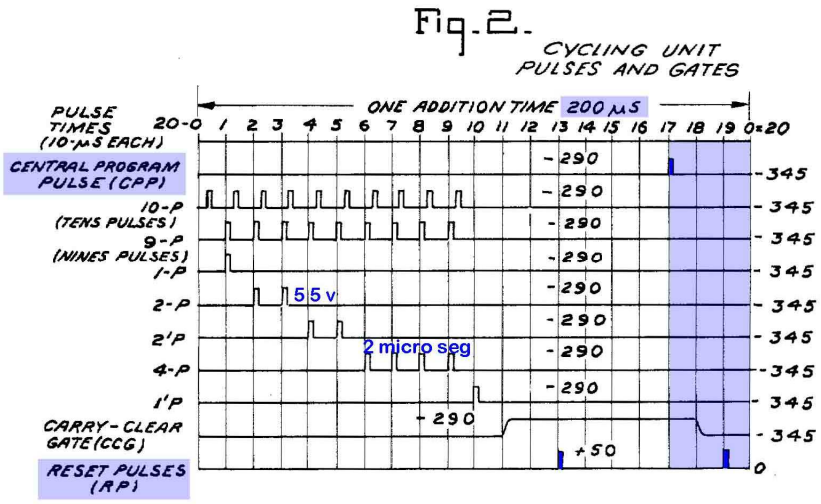


Imagen 21 –Los pulsos del ENIAC en un ciclo de suma

El programador maestro que se muestra en la imagen 19 y 22, con sus controles en la imagen 23, estaba formado por dos módulos; era un elemento muy útil que servía para desencadenar una secuencia de operaciones y coordinaba los acumuladores, permitiendo realizar bucles anidados. Cada módulo tenía un interruptor de encendido, *Heaters,* cuatro interruptores de asociación de década, diez interruptores de décadas (del 0 al 9) por seis etapas o fases y cinco interruptores de pasos, *stepper*, (del 1 al 6), con ellos se fijaba el número de veces a realizar una secuencia de órdenes. Conseguía por hardware resultados similares a una instrucción *for* actual.

Imagen 22 – Vista de la unidad de inicio, ciclos y el programador maestro

El programador maestro cada vez que recibía un pulso tipo *CPP*, emitía un pulso de programa en el siguiente ciclo de suma que desencadenaba las operaciones en los elementos conectados, incrementaba su contador en uno, comparaba dicho contador con el número establecido por los interruptores y cuando coincidían detenía el flujo del programa. Al tener dos componentes se podían ejecutar bucles anidados pasando de uno al otro.

Esto permitía al ENIAC realizar bucles *for* anidados de tamaño fijo. Sin embargo no conseguía la bifurcación condicional con dos opciones según se

cumpliera una condición, para ello usaban la discriminación de magnitud, similar a un *if … else* actual, que se explica en el acumulador.

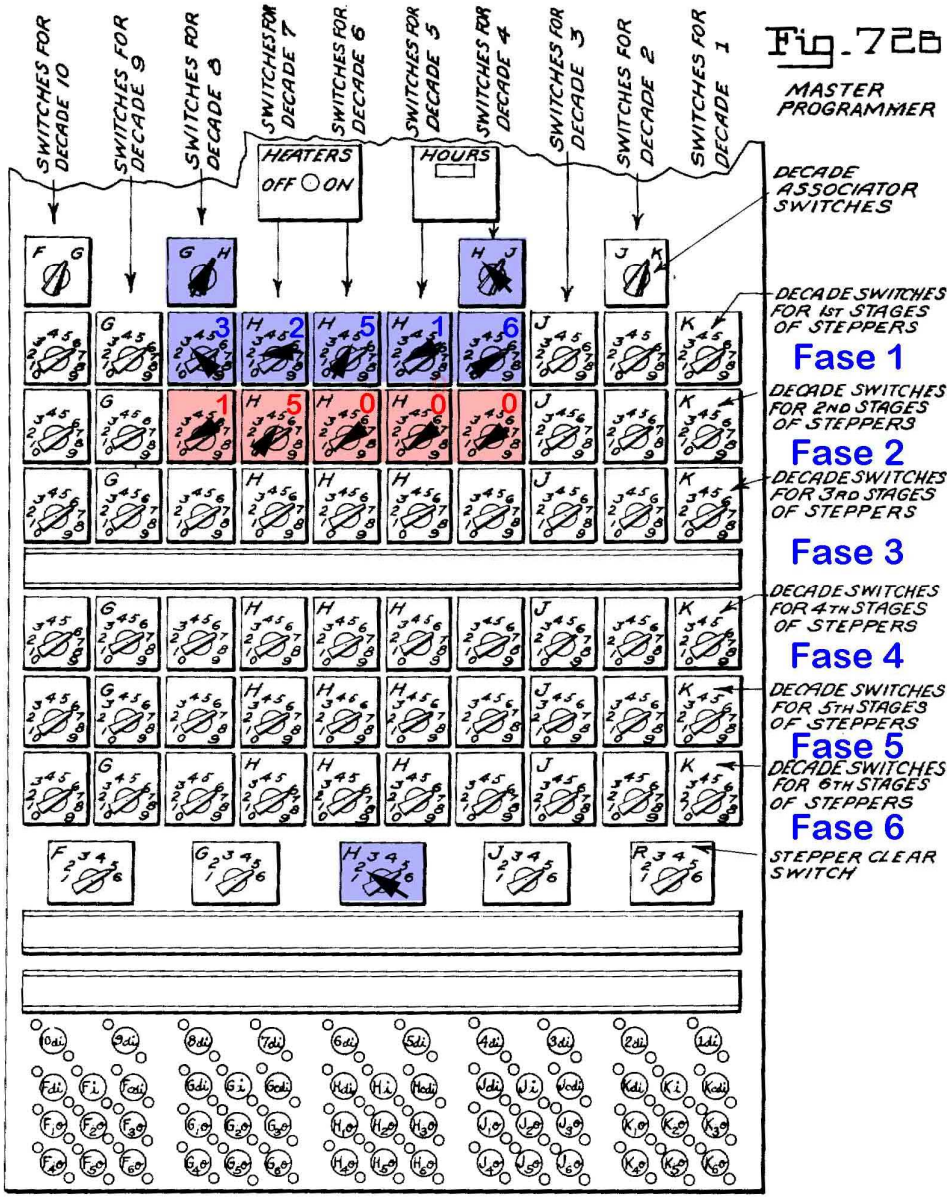


Imagen 23 – Detalle de los interruptores del programador maestro

4.4. Los buses

Aquí se modifica el orden en la exposición de la arquitectura pues conviene explicar los buses del ENIAC dado que es necesario conocerlos para abordar los demás componentes. Los buses de datos y de programa como se aprecia en la imagen 24, figura 38 de la patente, iban sobre bandejas apiladas, las de datos,

Digit trays por arriba, se numeraban en el simulador eniac.jar a partir del uno; las de programa, *Program Trays* por debajo, con una letra a partir de la A.

Los buses estaban divididos en tramos que daban servicio a cuatro componentes del ENIAC formando un grupo, de esta manera cada elemento tenía cerca su parte de los buses con las bandejas y tomas para conectarse de forma fácil. Además debían interconectarse los diez grupos para permitir que los pulsos viajaran de principio a fin del ENIAC por dichos buses. El bus de sincronización también iba en bandejas aunque de manera interna.

Por las bandejas pasaban las líneas troncales o *Trunks*, que eran dicho de manera sencilla, un cable múltiple de once líneas. Las líneas troncales eran iguales en el bus de datos y de programa, pero se diferenciaban en la toma conector. Por la parte alta en el bus de datos, había una toma por bandeja dado que se usaban las once líneas simultáneamente para transmitir o recibir, una del signo y diez para el número. Por debajo iba el bus de programa con once tomas, pues solo necesitaba una línea para mandar o recibir cada pulso de activación.

Las bandejas de datos se utilizaban muchas veces, pero solo debía transmitir uno aunque podían recibir varios. Había unos adaptadores que regulaban la transmisión, se podían utilizar acumuladores de desplazamiento para desplazar el número entre el transmisor y receptor, equivalía a multiplicar o dividir por una potencia de diez, también había acumuladores de borrado que suprimían los pulsos de ciertos lugares del número.

En cada bandeja del bus de programa, como el simulador, consideramos los conectores de línea numerados del 1 a 11 a partir de la izquierda; todas las líneas con el mismo número estaban conectadas internamente entre sí, por ejemplo, en la primera bandeja las líneas 5 de todos los componentes estaban unidas.

Los conectores eran individuales, por ello cuando un acumulador, por ejemplo necesitaba dos conexiones a la línea 4, había varias soluciones, conectarse a la suya y a la toma de la línea 4 libre más próxima o bien hacer un puente vertical en otro lugar, conectando las tomas de la línea 4 de la bandeja A y B; de esta forma en todo el bus de programa, esas dos líneas 4 pasaban a estar unidas. Este puente físicamente sería igual si se hace en horizontal con las líneas inmediatas, pero en el simulador eniac.jar se obtienen resultados diferentes.

Esto permitía el funcionamiento del ENIAC, que dicho de forma sencilla era lanzar un pulso de activación desde la unidad de inicio, recibirlo las unidades conectadas, ejecutar sus operaciones y al terminar enviar un pulso a la siguiente línea para desencadenar el mismo proceso y así sucesivamente. Pero esto se complicaba, pues lanzar un pulso a la línea de programa desencadenaba varias

operaciones por distintos elementos, apareciendo un problema al durar tiempos distintos, lo cual obligaba a planificar cuándo y por quien se lanzaba el pulso.

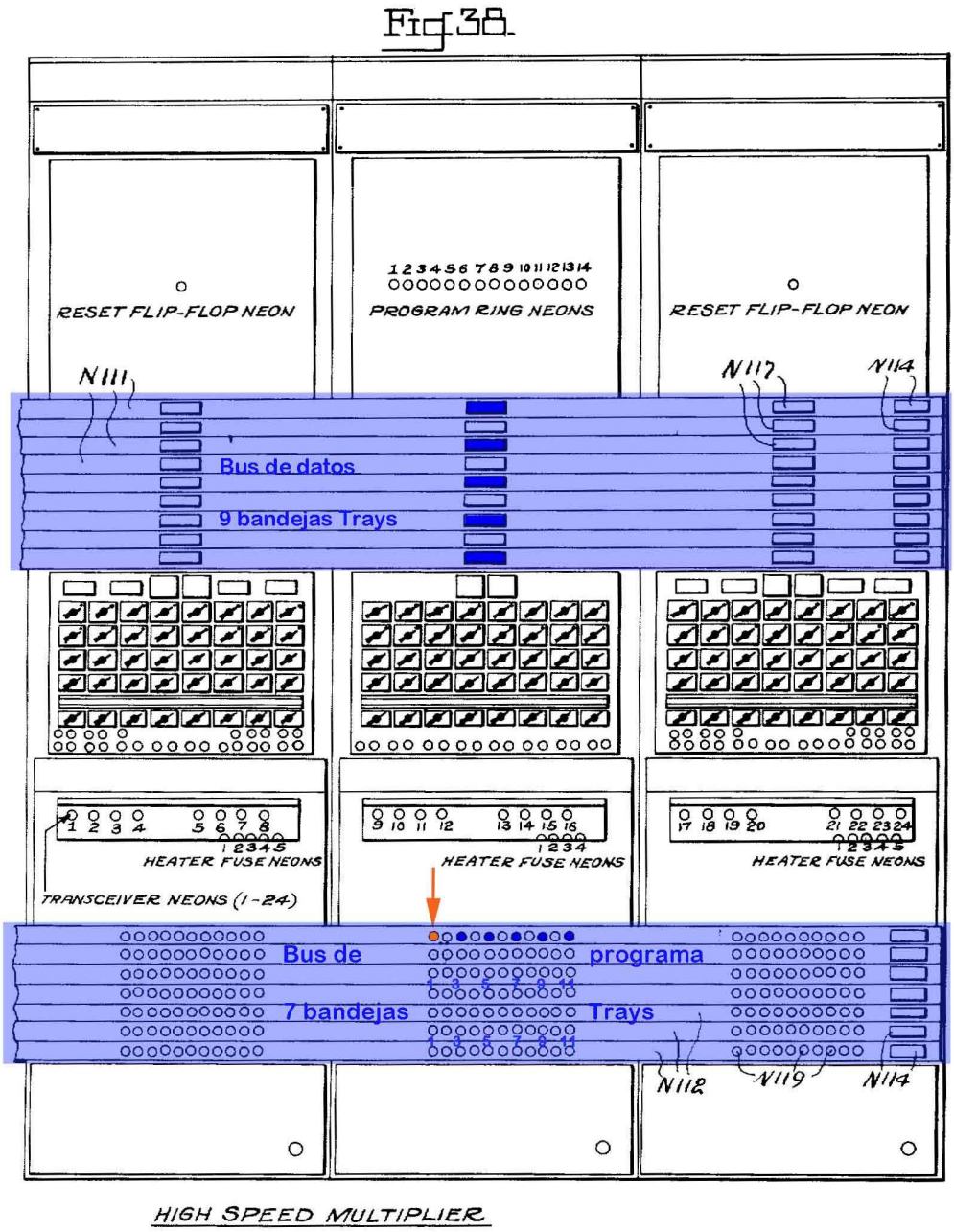


Imagen 24 – El multiplicador con los buses

El ENIAC permitía paralelizar pero no tenía mecanismos de control. Por ello el programador debía calcular previamente todas las duraciones para coordinar el correcto funcionamiento, determinando cuantos ciclos duraba cada una de las operaciones de cada etapa y desde cual de los elementos se mandaba la señal de operación terminada o pulso, para que comenzara la siguiente etapa. También había que controlar las líneas ya usadas para no crear un bucle.

Como resumen, para programar el ENIAC se partía de un problema de cálculo a realizar, que se materializa en una serie de ecuaciones matemáticas. Estas ecuaciones se organizaban en una secuencia de operaciones básicas que el ENIAC podía ejecutar; sobre esta secuencia se planificaba el recorrido y almacenamiento de los datos iniciales, resultados intermedios y finales.

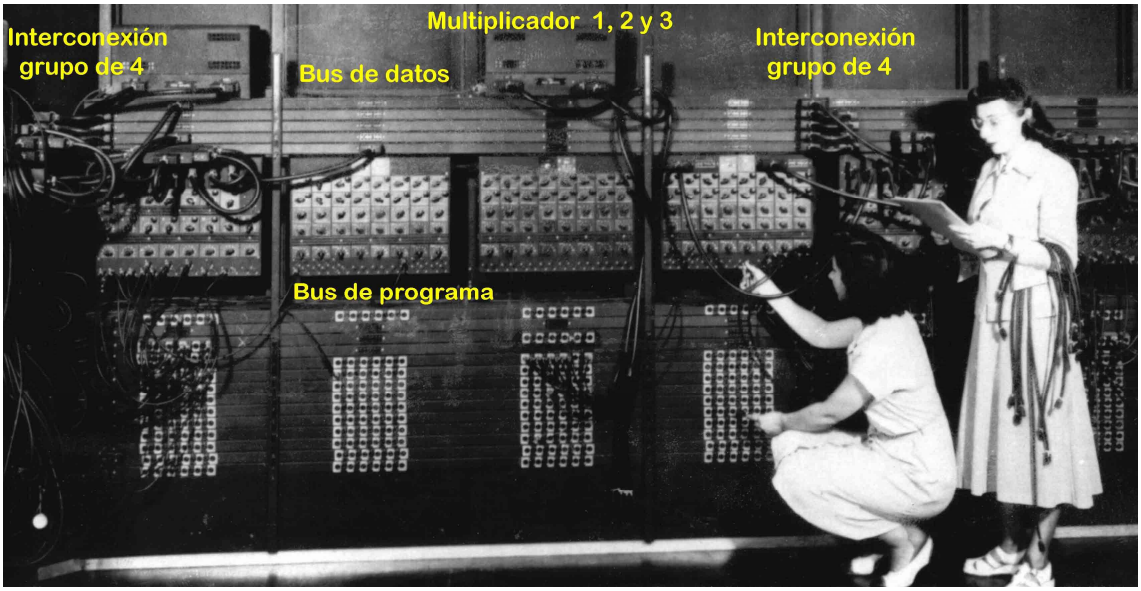


Imagen 25 – Vista del multiplicador y los buses

Una vez resuelto esto, como el ENIAC se programaba por etapas, había que repartir la secuencia de operaciones entre las etapas previstas. Al lanzar la primera etapa con el pulso inicial al bus de programación, estaban determinadas las operaciones, de la secuencia, que se conectaban a dicho pulso para ejecutarse. Como se podían paralelizar los recursos disponibles se había analizado antes cuando terminaba cada una y que elemento enviaba la señal de activación de la segunda etapa con un pulso a la siguiente línea del bus de datos.

Este diseño teórico obtenido a partir de la patente sirve para comprender el funcionamiento, pero no se cumplía exactamente en el ENIAC como se aprecia en la imagen 25. En el bus de programa debajo del multiplicador, hay conectores de datos, además tenían seis conectores por bandeja en lugar de once, y hay bandejas con tomas de datos y de programa. El ENIAC transmitía de manera general los números en serie, en forma de pulsos como se ha dicho y era más rápido. Pero también lo hacía estáticamente, con señales de estado estacionarias para conexiones entre el multiplicador, el divisor/raíz cuadrada y las tablas de funciones con sus acumuladores, aspecto que no se detalla.

4.5. La aritmética y el almacenamiento

Este apartado de la arquitectura tenía veinte acumuladores, el multiplicador de alta velocidad y el divisor/raíz cuadrada que le daban su potencia de cálculo. Los acumuladores eran los elementos más numerosos, formaban la estructura base, denominándose acumuladores porque sumaban a su contenido todo lo que recibían, pero prácticamente eran un registro acumulador, muchos de ellos ya asignados, seis al multiplicador, siete al divisor/raíz cuadrada, etc.

4.5.1. Los acumuladores

Generalidades, como se ve en la imagen 28, figuras 31 y 32 de la patente, cada acumulador cumplía varias funciones, la aritmética de acumular, cada vez que recibían un número lo sumaban al existente, similar a una ALU. Otra era el almacenamiento del resultado, análogo a un registro. Además tenían una unidad de control local para programarse si recibía o emitía ya fuera el número almacenado o su contrario. También disponía de circuitos de entrada/salida con sus tomas para conectarse a las bandejas del bus de datos y de programa.

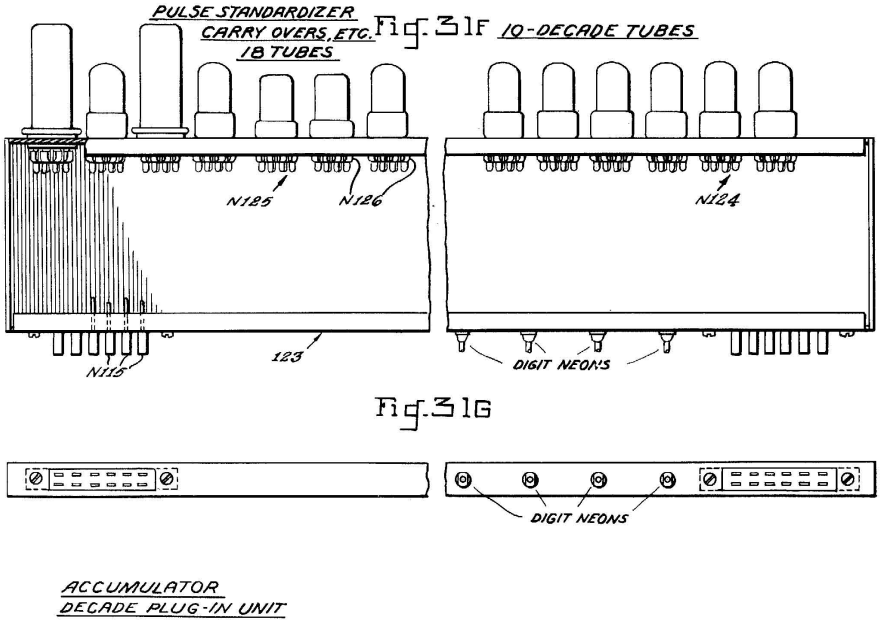


Imagen 26 – El contador de décadas

Los acumuladores, imagen 28, tenían en la parte superior un panel con diez décadas, los *Blinken ligths* o luces destelladas; cada década tenía diez bombillas de neón para encender la correspondiente al dígito almacenado, del cero al nueve; además había un indicador P/M, *Minus,* que se encendía según fuera el signo del número. Debajo había diez pilotos de neón de los biestables de década.

Cada década por detrás, tenía su **contador de décadas**, mostrado en la imagen 26 figura 31G de la patente y la fotografía de la imagen 27, era un módulo con 28 válvulas que guardaba uno de los dígitos, mediante un anillo contador de diez posiciones con sus correspondientes biestables, todos puestos a cero salvo el del valor almacenado que estaba a uno; así la década que almacenaba un tres, tenía a uno el biestable cuatro.

El funcionamiento del ENIAC era mediante representación decimal, esto fue estudiado descartando hacerlo binario porque en cada acumulador los diez contadores de décadas usaban 280 válvulas, frente a las 450 necesarias con un sistema binario de 30 bits para representar el mismo intervalo de números, lo cual disminuía la fiabilidad. Para los números negativos no cambiaban el signo pues los representaban en complemento a 10, este causaba menos problemas con el redondeo y la eliminación de cifras significativas que el complemento a 9.

Un número negativo en complemento a 10 se calculaba restando cada dígito de nueve, sumando al final un uno y poniendo el signo en M, *Minus*. El signo valía *uno* para números negativos y *cero* en otro caso. El indicador P/M como el de acarreo, se hacía efectivo con el segundo pulso de *Reset* en el siguiente ciclo.

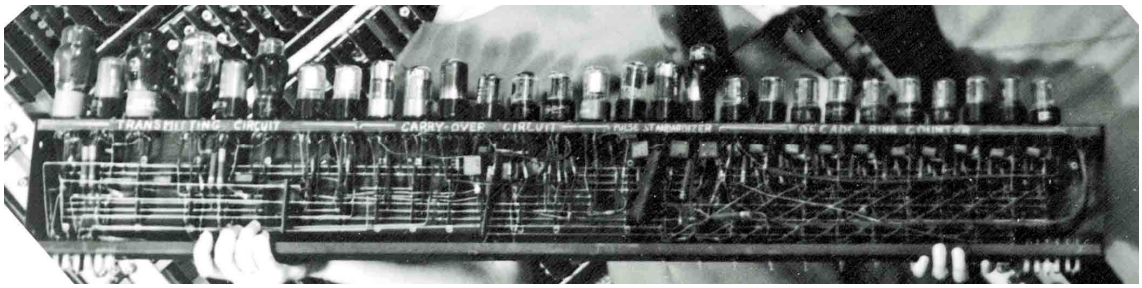


Imagen 27 – Detalle del contador de décadas

El proceso para sumar en un acumulador otro número que le llegaba, explicado de manera simplificada era el siguiente. Comenzado por la década unidades, más a la derecha, se recorría el anillo contador, a partir del dígito inicial almacenado, esto es el dígito unidades del acumulador que correspondía al biestable puesto a uno, este se ponía a cero, se desplazaban tantas posiciones como valía el dígito unidades a sumar y se ponía a uno el biestable final, cuando en este desplazamiento se producía un desbordamiento, (por pasar del 9 al 0) se generaba un digito de acarreo que se enviaba a la década siguiente, decenas.

En la siguiente década, decenas, se repetía el proceso incluyendo caso de haberlo el acarreo y así hasta acabar la década diez. En la posición once un contador binario P/M fijaba el signo, *cero* positivo con cero pulsos, *uno* negativo con nueve y con un número impar de pulsos se cambiaba el signo. El proceso era más complejo pues los acarreos y el signo se hacían efectivos por el *Reset* en

el siguiente ciclo, pero además los dígitos se transmitían con los trenes de pulsos 9-P, 10-P y 1´P, y el signo tenía su procedimiento, por ello no se va profundizar.

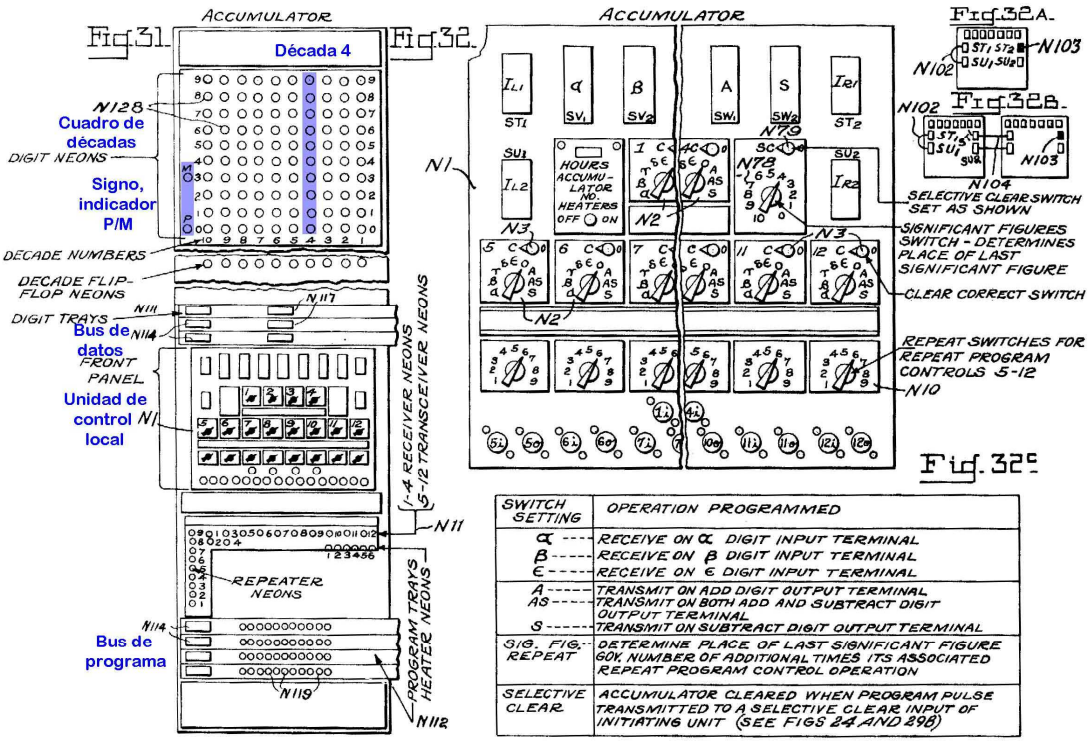


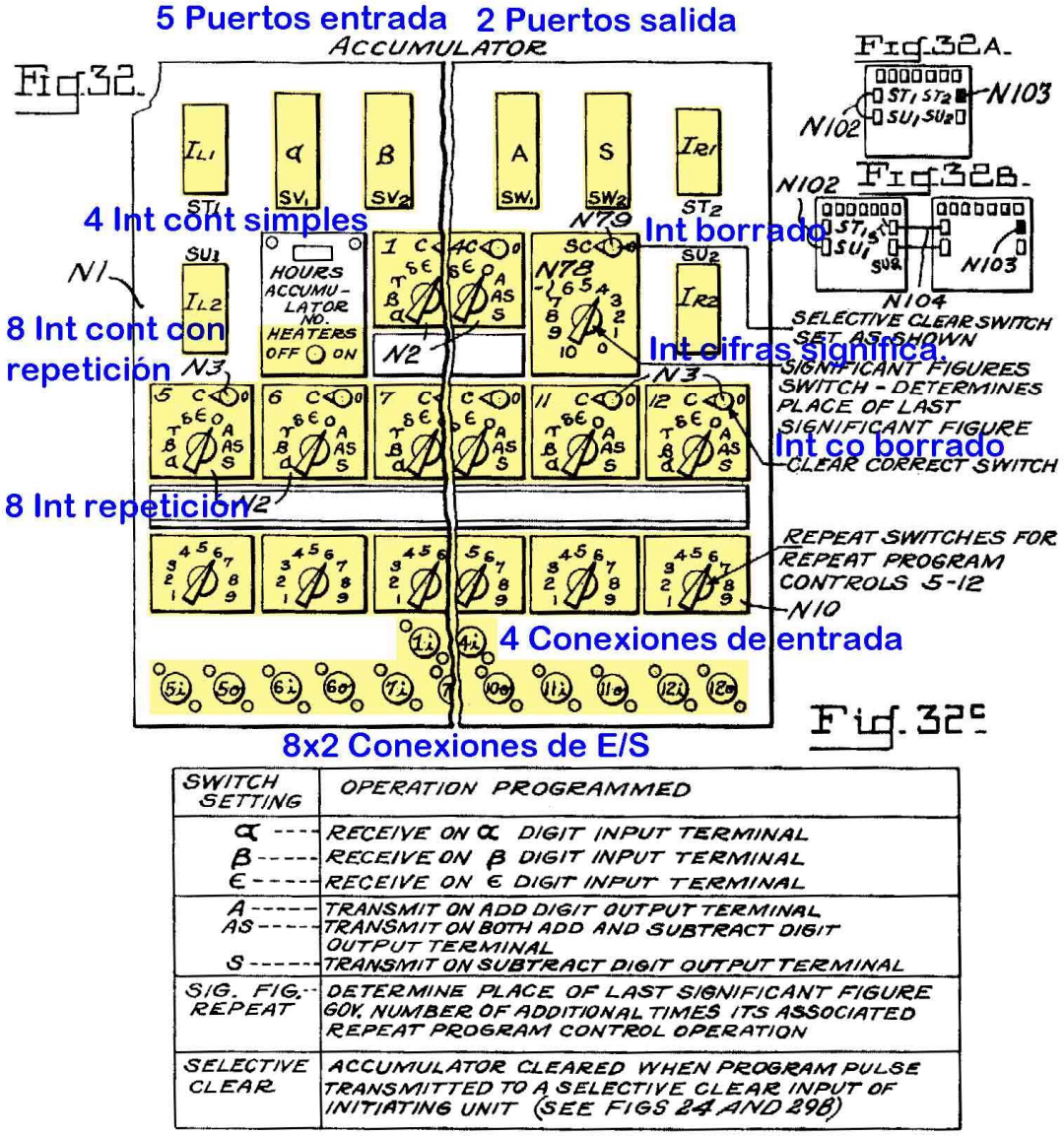
Imagen 28 – El acumulador y su unidad de control

La unidad de control local detallada en la imagen 29, figura 32 de la patente, estaba debajo del cuadro de décadas, *Blinken ligths,* entre las bandejas del bus de datos y el bus de programa, tenía dos grupos de entradas y salidas para conectarse a dichos buses. Permitía programar el acumulador, fijando los interruptores y cableando las conexiones.

Como se viene haciendo, la imagen 29 lleva marcado en amarillo las partes simuladas por el *applet* de Java eniac.jar. Estas imágenes comentadas son una explicación gráfica del funcionamiento del ENIAC, en ocasiones mejor que el texto, recogen de forma visual detalles que por escrito no se alcanzan y se utilizan para poder profundizar, sin alargar el documento.

Cada acumulador en su unidad de control tenía un interruptor de encendido, *Heaters*, disponía de cinco puertos de entrada denominados alfa, beta, gamma, delta y épsilon y dos puertos de salida A para enviar el valor del número acumulado y S para enviar el valor cambiado de signo, en complemento a 10. Los siete puertos de entrada y salida se conectaban al bus de datos mediante un cable con toma única de once hilos, para enviar o recibir números.

Debajo de los puertos había doce interruptores de control de programa, que permitían realizar hasta doce operaciones independientes. Distribuidos en dos filas, en la primera había cuatro interruptores de control simples, que solo podían realizar una operación y no generaban pulso de salida; en la siguiente fila había ocho interruptores de control que tenían debajo sus correspondientes interruptores de repetición, de una a nueve veces. Así estos ocho interruptores podían reiterar su operación según se fijara en dicho interruptor de repetición.



SWITCH SETTING	OPERATION PROGRAMMED
α ----	RECEIVE ON α DIGIT INPUT TERMINAL
β -----	RECEIVE ON β DIGIT INPUT TERMINAL
ε -----	RECEIVE ON ε DIGIT INPUT TERMINAL
A -----	TRANSMIT ON ADD DIGIT OUTPUT TERMINAL
AS ----	TRANSMIT ON BOTH ADD AND SUBTRACT DIGIT OUTPUT TERMINAL
S ------	TRANSMIT ON SUBTRACT DIGIT OUTPUT TERMINAL
SIG. FIG.-- REPEAT	DETERMINE PLACE OF LAST SIGNIFICANT FIGURE GOV. NUMBER OF ADDITIONAL TIMES ITS ASSOCIATED REPEAT PROGRAM CONTROL OPERATION
SELECTIVE CLEAR	ACCUMULATOR CLEARED WHEN PROGRAM PULSE TRANSMITTED TO A SELECTIVE CLEAR INPUT OF INITIATING UNIT (SEE FIGS 24 AND 29B)

Imagen 29 – La unidad de control del acumulador

Los doce interruptores de control tenían en su esquina superior derecha otro interruptor, el de correcto borrado con dos posiciones, la cero y la C donde se activa, de tal modo que cuando le llegue un pulso de activación estando fijado

para recibir, recogerá un pulso 1´P. El resultado es que va añadiendo un uno al acumulador cada vez que recibe un pulso de activación, por procesar el 1´P.

Los cuatro interruptores de control simples solo tenían toma de entrada para ser activados; los otros ocho interruptores de control tenían cada uno su conexión de entrada y de salida, con las cuales al concluir las operaciones, este acumulador podía generar el pulso de salida para activar la siguiente etapa. Se indica que en cada etapa al programar cableando, solo uno de los componentes implicados debía generar el pulso de salida para arrancar la siguiente.

Todos los interruptores tenían sus nueve posiciones con la siguiente distribución, cinco posiciones para los puerto de entrada, una posición cero para no hacer nada durante un ciclo de suma, similar a una pausa y tres posiciones con los puertos de salida, A para transmitir el numero, S para su contrario, por ello en el ENIAC restaba el número si se enviaba por este puerto S, o bien la tercera AS que enviaba por los dos a la vez. Cada interruptor fijado para una operación, con sus correspondientes conexiones al bus de datos y al de programa era similar a una instrucción de programa actual.

El modo de programar un acumulador con un ejemplo, podría ser poner el interruptor en enviar AS. Para poder recibir el pulso de activación e iniciar la operación había que conectar un cable entre el bus de programa en una línea y la entrada del interruptor. También había que conectar las salidas A y S del emisor a dos bandejas de datos. Luego se programaban los receptores.

El que recibía A, colocaba el interruptor de control para recibir, por ejemplo en el puerto alfa, se conectaba a la bandeja de datos por la que enviaban A y se conectaba al bus de programa en la misma línea que el emisor. Se repetía la operación con el receptor de S. Hecho esto se podría ejecutar la etapa, pero habría que determinar desde que acumulador se enviaba el pulso de activación a la siguiente etapa, conectando su salida a la siguiente línea de programa. Como se ve la programación era bastante prolija y ello sin complicarla con repeticiones ni operaciones de distintas duraciones.

A la derecha de los cuatro interruptores de control simple había un doble interruptor, arriba estaba el borrado selectivo del acumulador, que fijándolo en C se realizaba al pulsar manualmente el botón de borrado, desde la unidad de inicio. Debajo estaba el interruptor de cifras significativas de los números (de 10 a 0), que permitía fijarlas. Pero cuando se ponía un número menor de 10, en la primera posición decimal se colocaba un 5 en lugar de una coma. Se ha usado con el simulador probando 7 cifras significativas, se introdujo *un 10* (10500), después *de sumar y restar un 3* (3500), *resultó 14000 y 7999* que confunde.

Manejo de números con veinte dígitos, el ENIAC usaba normalmente diez dígitos configurado según se ve en el acumulador derecho de la imagen 30, con sus *interconectores* colocados de ese modo, los *Il1 e Il2* conectados entre sí y el *Ir1* solo. Para operar con veinte dígitos se disponían como los dos acumuladores de la izquierda en la imagen 30, conectándose en pareja, el izquierdo tenía sus *interconectores Il1 e Il2* conectados entre sí, pero los *Ir1 e Ir2* se conectaban al *Il1 e Il2* del acumulador derecho y el *Ir1* derecho suelto.

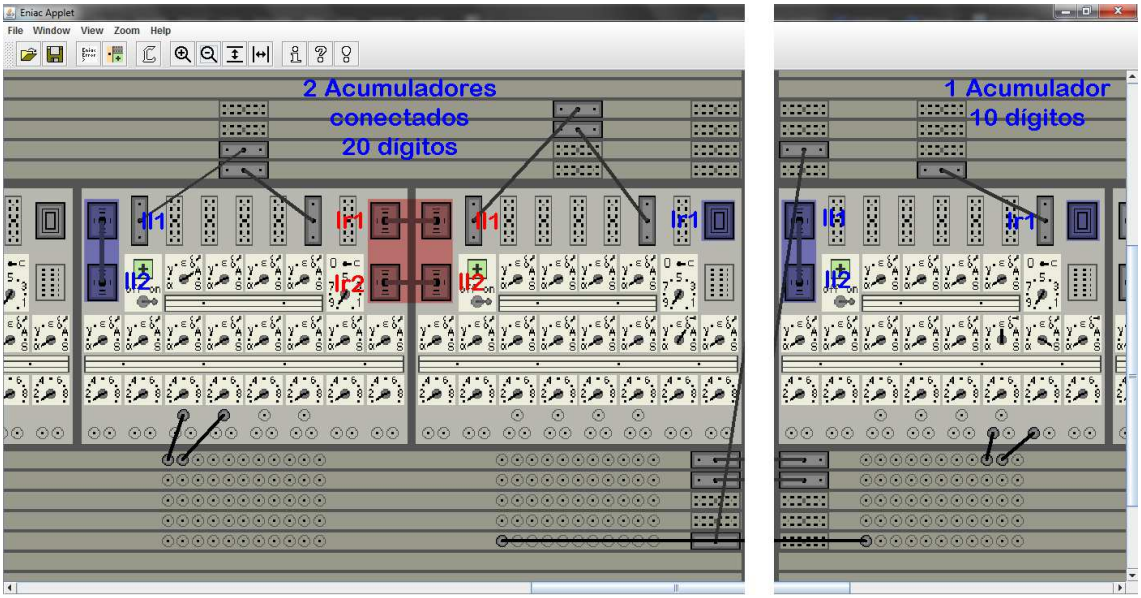


Imagen 30 – Modos de conexión de los acumuladores

Además ambos acumuladores tenían sus conexiones al bus de datos aunque compartían la del bus de programa. El signo se consideraba el del acumulador izquierdo y los acarreos pasaban de la última década del derecho a la primera del izquierdo. Había otras cuestiones de coordinación que no se tratan, porque para manejar veinte dígitos se necesitan dos ciclos de suma.

Discriminación de magnitud, al tratar el programador maestro en el apartado 4.3 se expuso que podía hacer bucles *for* de tamaño fijo pero no podía realizar una bifurcación condicional o instrucción *if … else* según se cumpliera una condición. Esto lo conseguía el ENIAC con la discriminación de magnitud, que es un recurso para comparar dos cantidades restándolas y usar el resultado negativo para obtener del indicador de signo, un pulso de programa. La discriminación de magnitud se explica detalladamente porque es la estructura sobre la que se asientan casi todos los programas hechos para el ENIAC.

Recapitulando sobre los trenes de pulsos de la unidad de ciclos en el mismo apartado 4.3, se observa que prácticamente todos los trenes de pulsos eran combinaciones del mismo pulso básico, uno de dos microsegundos y cincuenta y

cinco voltios a cien kilohercios. Había distintos trenes de pulsos por tener más o menos pulsos individuales y en instantes diferentes. Pero la uniformidad del pulso básico era una cuestión importante, porque al ser iguales, en principio se podrían usar en cualquier lugar, bus de datos o de programa, e incluso se podría pasar del bus de datos al de programa como veremos.

El otro factor a considerar es el indicador de signo P/M, se ponía a uno la línea 1 del bus de datos cuando el número era negativo y además estaba en cero si era positivo o cero; esto dicho de manera sencilla suponía que con un uno en la línea 1, se producía una corriente que podría ser usada como pulso de activación, pero el pulso se producía en el bus de datos que realizaba otra tarea, usar las once líneas para transmitir números, no para activar elementos.

Hasta aquí se tiene que podíamos comprobar la condición menor que "b", con cualquier número "a," restándole el valor de "b"; se cumplía que era menor, o resultado negativo, cuando había un uno en línea 1, que además podría servir de pulso de activación. Para ello se enviaba el dato por el puerto A al bus de programa, conectando luego su línea 1 a la línea que correspondiera del bus de programa y así se conseguía un pulso de activación. La condición mayor o igual se lograba enviando el dato negado, por el puerto S. No se podía comparar solo la igualdad, porque no se diferenciaba el cero del resto de valores positivos.

Para facilitar la comprensión se va a emplear un programa con un algoritmo que calcula el producto usando una instrucción de bifurcación condicional o *if*, cuyo código lógico figura debajo, mostrando a continuación su implementación con el simulador eniac.jar, lo cual nos permitirá observar mejor los detalles de la discriminación de magnitud sobre este ejemplo práctico.

```
/** Calcula el producto de M por N */
  Acum1= 1, Acum2= M;          //Comienzo, pulso a línea 1
  Acum3= N;                    //Pulso a línea 2
  if(Acum3 >= 0){              //Pulso a línea 3
                               //Espera
  Acum4+=Acum2, Acum3-=Acum1; //Pulso línea 4 y pulso línea 3
  }                            //Espera
  Acum4-=Acum2, Acum3+=Acum1; //Pulso a línea 5, final
```

La imagen 31 es la implementación de este producto realizado con el simulador eniac.jar, aplicación que se explica en el apartado 5 de programación. Se parte de lo siguiente, el valor en el tercer acumulador es mayor o igual que cero y las bandejas de programación A y B tienen sus líneas puenteadas verticalmente, así la línea 3 de A y de B están conectadas. Ahora solo interesa identificar el proceso sobre la imagen 31. Se inicia *en amarillo*, por el 1 en la línea 3, es la etapa de comparación dura los números 2, 3 y 4 que activa la línea

4 con la siguiente etapa *en rojo*, las operaciones del producto y acaban en 5 que cierra el bucle, enviando otro pulso de activación a la línea 3 y se repite el ciclo.

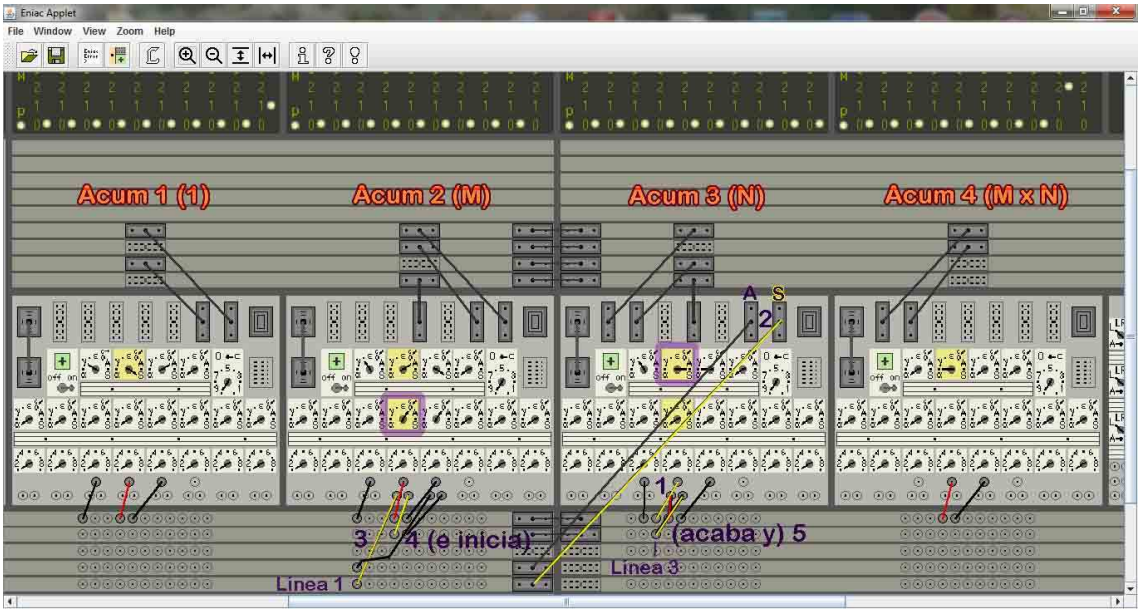


Imagen 31 – La discriminación de magnitud en un programa

Con más detalle siguiendo los números, *uno*, el tercer acumulador recibe un pulso de activación por la línea 3 del bus de programa que arranca la etapa de comparación *if*, en amarillo; *dos*, el tercer acumulador envía su valor almacenado por A y S al bus de programa. Mientras dicho valor sea mayor o igual que cero, el flujo enviado por S dará el pulso de activación; *tres*, el segundo acumulador espera sin hacer nada un ciclo para que sea efectivo el indicador de signo, se suele llamar programa simulado o *Program dummy,* que sirve para aislar un pulso. Y *cuatro*, obtiene del flujo S, el pulso enviándolo a la línea 4.

Este pulso de activación desencadena la siguiente etapa, en rojo, el conjunto de operaciones del producto, todas a la vez, paralelas y de un solo ciclo de suma. Entre las operaciones, el tercer acumulador recibe un menos uno del primer acumulador, le resta uno. Una vez acabada la etapa, *cinco*, es el tercer acumulador quien envía un pulso, pero a la línea 3 de la etapa anterior, cerrando el bucle que se repite, lanzando el tercer acumulador su valor por A y S hasta la finalización, producida cuando este acumulador alcance un valor negativo y se continúe por el otro camino. Entonces será el flujo de A quien proporcione el pulso, lanzándolo por la línea 5, que activará la etapa final.

Por las peculiaridades del ENIAC, el signo no se hacía efectivo en su ciclo como ha señalando, la resta se calculaba en el ciclo, pero los acarreos y el signo los hacía efectivos en el siguiente ciclo. Ello suponía realizar la comparación restando, esperar una pausa de un ciclo y continuar el programa según se

cumpliera o no la condición. En esta explicación para hacerla más sencilla se está señalando que con el signo negativo se pone el indicador a uno, pero esto como se ha explicado en la unidad de ciclos del apartado 4.3, sucedía cuando se transmitían nueve pulsos de dígitos al cable de signo.

Como se ha podido comprobar esto era un poco más complejo; en principio requería bastante cableado, usar dos bandejas de programa aisladas para recibir desde el bus de datos y conectar su línea 1 a la etapa que correspondiera en la bandeja del programa en curso. Además las bifurcaciones condicionales solían estar dentro de un bucle de operaciones para multiplicar en este caso, que se organizaba cableando para recorrer un camino con una serie de instrucciones mientras se cumpliera la condición, de tal manera que al final cuando no se cumplía, se continuaba por el otro camino, ejecutando el resto del programa.

4.5.2. El multiplicador de alta velocidad

El multiplicador de alta velocidad ocupaba tres módulos del ENIAC, como se puede ver en la imagen 24 y 32; para operar con diez dígitos necesitaba cuatro acumuladores, el del multiplicador *´Ier*, el multiplicando *´Icand*, el acumulador izquierdo del producto parcial, *LHPP–I* y el acumulador derecho del producto parcial, *RHPP-I*. Sin embargo al multiplicar dos números de diez dígitos el resultado tiene entre dieciocho y veinte dígitos, con lo cual necesitaba a menudo otros dos acumuladores, el *LHPP–II y RHPP-II* para almacenar el resultado completo.

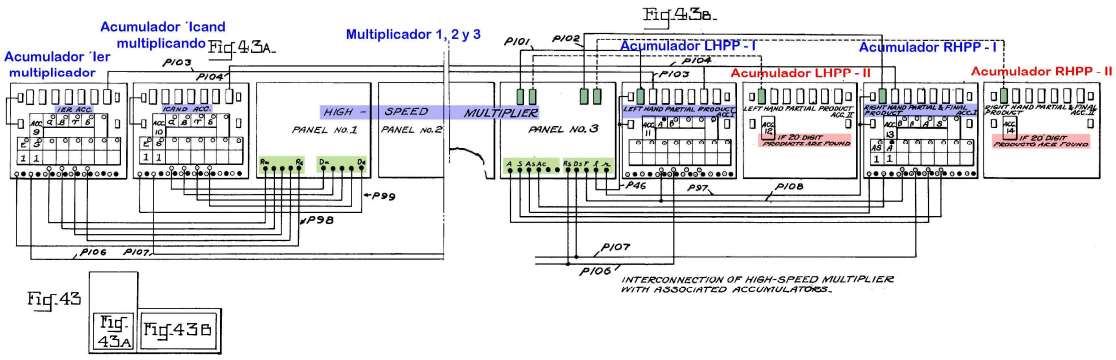


Imagen 32 – Configuración del multiplicador y sus seis acumuladores

El producto se hacía multiplicando con unas tablas directamente cada dígito del multiplicador *´Ier,* por el multiplicando *´Icand*, guardando los resultados en los acumuladores productos parciales izquierdo y derecho, *LHPP–I* y *RHPP–I*. La multiplicación de un número de diez dígitos por otro con d dígitos en el multiplicador, era bastante eficiente pues solo necesitaba d más cuatro ciclos de suma, de doscientos microsegundos cada uno.

En el producto se usaban las conexiones de datos en serie o por pulsos, vistas en los acumuladores. Pero también se usaban conexiones estáticas, que en el esquema de la imagen 32 se observan por la parte inferior. Dichas conexiones estáticas se establecían entre los acumuladores *´Ier* e *´Icand* con el multiplicador y los acumuladores de resultados parciales *LHPP-I* y *RHPP-I*, para transmitir y recibir los números con el signo.

La transmisión estática de números con señales de estado estacionarias requería cuatro cables para transmitir los diez posibles valores. Este tipo de conexiones se usó entre el multiplicador de alta velocidad, el divisor/raíz cuadrada, las tablas de funciones y la impresora, con las salidas estáticas de los biestables de los contadores de décadas, en los acumuladores asignados.

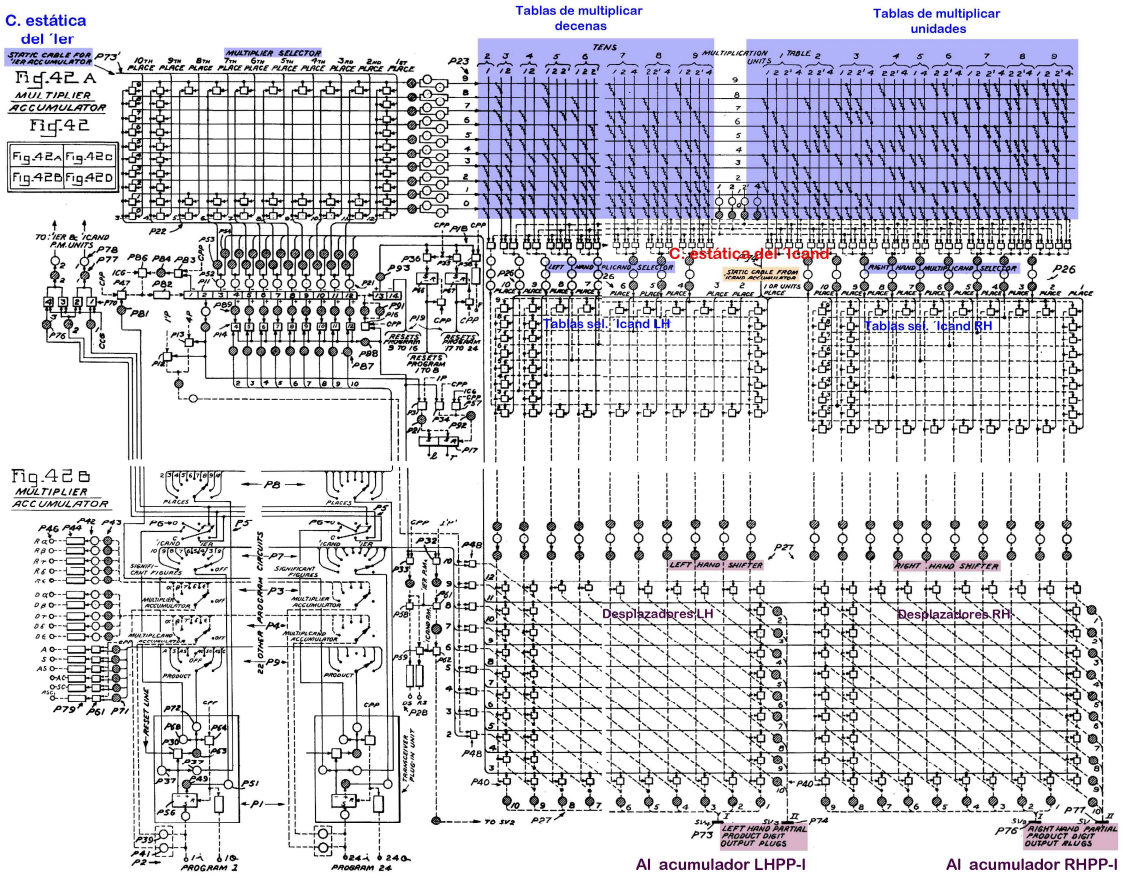


Imagen 33 – Recorrido de los circuitos internos del multiplicador

El recorrido efectuado por un producto se muestra en la imagen 33 sobre los circuitos internos del multiplicador para facilitar su seguimiento. El multiplicador de alta velocidad comenzaba recorriendo las cifras del multiplicador *´Ier* con la tabla selectora del multiplicador, para elegir las tablas a emplear correspondientes a sus dígitos.

Había dos tablas, la izquierda de decenas *Ten*, del dos al nueve y la tabla derecha de unidades *Units*, del uno al nueve, donde una vez seleccionadas las adecuadas a los dígitos del multiplicador *´Ier*, estas proporcionan directamente los resultados a partir de los dígitos del multiplicando. El producto desde la tabla decenas y la tabla unidades se realizaba por dos caminos independientes.

De las tablas se llegaba a sendos acumuladores parciales donde con la tabla de selección *´Icand* izquierda, *Left Hand ´Icand Selector Table,* efectuaba el producto del multiplicando *´Icand* por la tabla de multiplicar decenas o bien con la tabla de selección *´Icand* derecha*, Right Hand ´Icand Selector Table* el de las unidades. Así se conseguía mucha rapidez pues en cada ciclo de suma calculaba el producto de todo el multiplicando por un dígito del multiplicador.

De aquí continuaban atravesando los desplazadores *Left Hand Shifter* y el *Right Hand Shifter* que colocaban los productos parciales en sus posiciones correctas para el envío, las decenas un lugar a la izquierda de sus correspondientes unidades. Una vez situados, los resultados se enviaban fuera del multiplicador a los acumuladores parciales.

De esta manera, trabajando con diez cifras, los dígitos de las tablas de decenas llegaban al acumulador del producto parcial izquierdo, *LHPP-I* y los dígitos de las unidades al acumulador del producto parcial derecho, *RHPP-I*. Cuando el resultado tenía veinte dígitos había dos acumuladores más, ver imagen 32, *LHPP-II* y *RHPP-II* que permitían recogerlos completos.

En el multiplicador cuando algún elemento era negativo aplicaban un factor de corrección antes de sumar los productos parciales. Una vez terminados los productos de los d dígitos del multiplicador *´Ier* por el multiplicando *´Icand*, el resultado final se obtenía reuniendo los resultados de los dos acumuladores parciales en uno; cuando usaban veinte dígitos se realiza un proceso análogo con los cuatro acumuladores parciales, llevándolos a dos acumuladores.

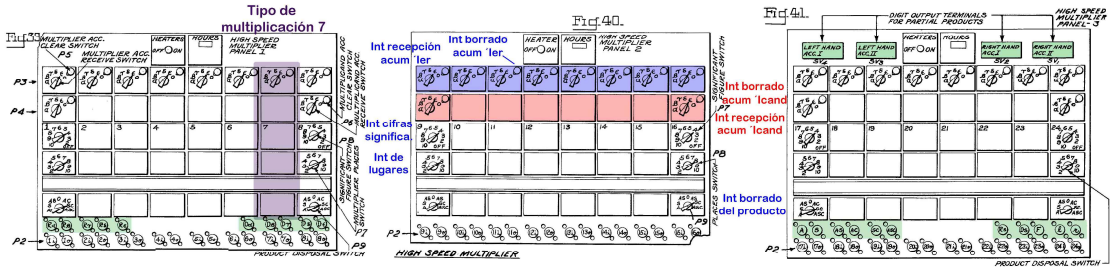


Imagen 34 – La unidad de control del multiplicador

La unidad de control, el manejo del multiplicador se hacía con los controles que se muestran en la imagen 34, las veinticuatro columnas permitían

realizar veinticuatro tipos diferentes de multiplicaciones en un programa y cada una se podía utilizar varias veces, pero manteniendo la configuración.

En el panel central dos, se han señalado los interruptores principales de cada configuración o columna. Había dos interruptores de recepción similares, arriba para el multiplicador *´Ier* y debajo para el multiplicando *´Icand*. Cada interruptor doble tenía a la izquierda un interruptor con cinco posiciones para cinco entradas, alfa a épsilon y cero para apagado. En la parte derecha arriba, estaba un interruptor de borrado del acumulador al finalizar la multiplicación.

Debajo estaban el interruptor de cifras significativas que se fijaban para redondear el resultado, el interruptor de lugares fijaba cuantos lugares del *´Ier* se multiplican por el *´Icand* y el interruptor de borrado del producto, con siete posiciones, permitía transmitir el resultado de forma análoga a los acumuladores, A, S y ambos AS; en dos modalidades, sin y con borrado. Además había una posición cero para recuperar el producto final desde el acumulador, fijándole un interruptor de control y mediante un pulso de activación.

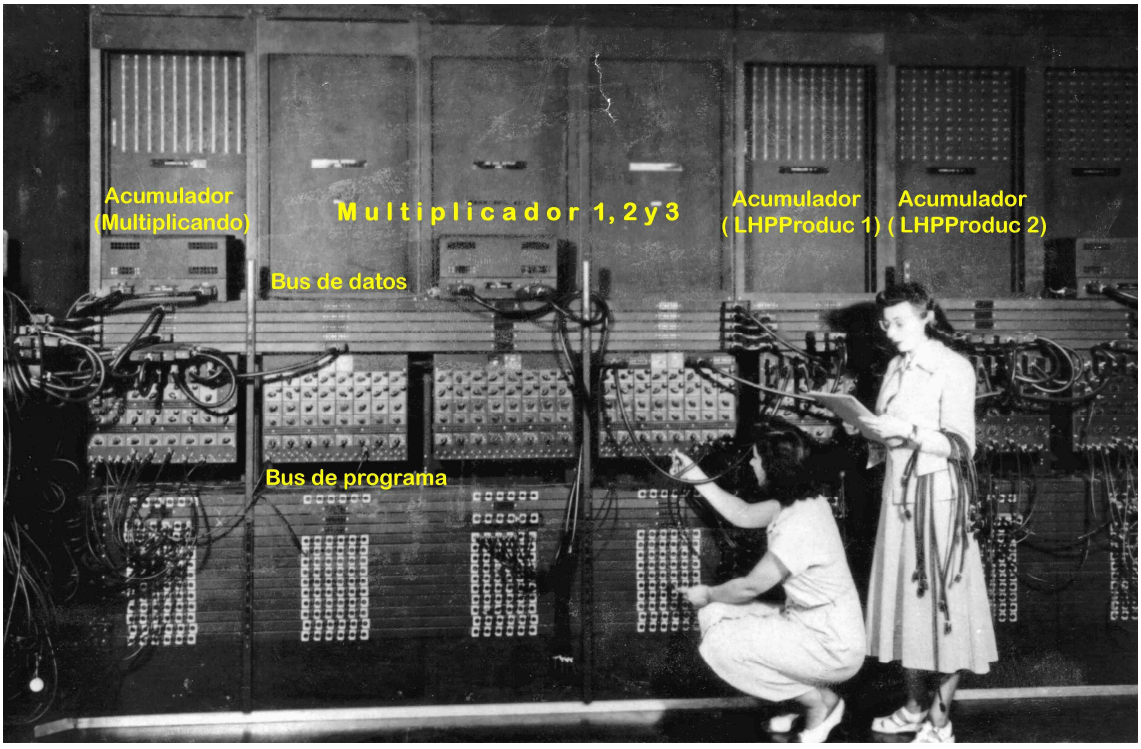


Imagen 35 – Vista general del multiplicador con tres acumuladores

El multiplicador como se mostró en la imagen 24 tenía una disposición parecida a los acumuladores en relación a los buses de datos y programa que lo recorrían por encima y debajo, aunque sus conexiones de entradas y salidas eran estáticas y un poco más complejas como se observa en la imagen 34, donde van señaladas en color verde, explicándose someramente a continuación.

Las entradas estáticas de datos desde los acumuladores eran diez en el panel uno, cinco del multiplicador *´Ier* denominadas *R* y cinco del multiplicando *´Icand* denominadas *D*. Las salidas de datos estaban en el panel tres, situadas en la parte inferior izquierda, eran seis, con o sin borrado para la opción A, S y AS. Además en esta parte inferior, a la derecha había otro grupo de cinco tomas Rs, Ds, F, l, r, para conectar entre el multiplicador y los acumuladores cuyo detalle se puede seguir en la imagen 32.

También en el panel tres, situadas en la parte superior, había cuatro tomas conexiones de datos, para enchufarse a los cuatro acumuladores de resultados parciales, *LHPP* y *RHPP*. Para más claridad se muestra una fotografía en la imagen 35 que contiene una vista general del multiplicador, donde se deduce que trabajaban con diez dígitos, pues solo conectan dos de estos acumuladores.

4.5.3. El divisor / raíz cuadrada

Generalidades, el módulo de la división/raíz cuadrada no era tan eficiente como el multiplicador al no disponer de tablas para realizar directamente las operaciones, era un elemento de control para dirigir sus siete acumuladores asociados cuando ejecutaban los algoritmos de la división o la raíz cuadrada.

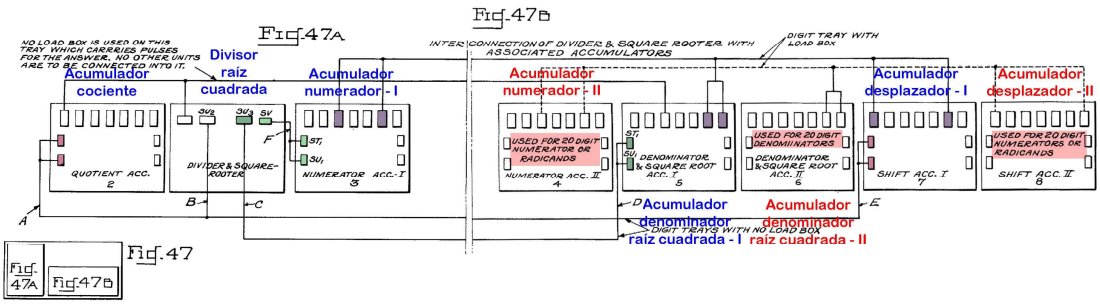


Imagen 36 – Configuración del divisor, raíz cuadrada y sus siete acumuladores

El divisor / raíz cuadrada cuya configuración se puede ver en la imagen 36, ocupaba un componente del ENIAC el cual, con diez dígitos empleaba cuatro acumuladores, el del cociente, el numerador-I, el denominador y raíz cuadrada-I y el acumulador desplazador-I. Como sucedía en el multiplicador, podía operar con veinte dígitos para lo cual usaba tres acumuladores más, el numerador-II, el denominador y raíz cuadrada-II y el acumulador desplazador-II que se emparejaban con sus análogos para diez dígitos.

De forma global *el divisor/raíz cuadrada* con sus cuatro acumuladores tenían estos cometidos generales. *El numerador* almacenaba el dato inicial,

denominado dividendo o radicando sobre el que se operaba; *el denominador y raíz cuadrada* proporcionaba el valor que se restaba al numerador, era un dato fijo o bien una serie de números impares variables en la raíz; *el cociente* guardaba el resultado de ambos, contabilizando el número de veces que se había efectuado la resta anterior. Además había un acumulador *desplazador* que permitía mover los datos de posición cuando los acumuladores los enviaban.

El divisor/raíz cuadrada usaba también conexiones estáticas con sus acumuladores, en la imagen 36 se puede observar el esquema que presenta algunas peculiaridades. La toma SV2 del divisor/raíz cuadrada estaba conectada al cociente por los interconectores izquierdos y al desplazador también en los interconectores izquierdos. Había una línea de datos entre la toma SV1 del divisor/raíz cuadrada con el cociente por el puerto alfa y el denominador por el puerto gamma. El numerador y el denominador usaban los interconectores izquierdos para conectarse a las tomas SV3 y SV del divisor/raíz cuadrada. Además mantenían una línea de datos entre el numerador, puertos gamma y A, el denominador, puertos A y S y el desplazador, puertos alfa y A.

Para la división y la raíz cuadrada se usaban algoritmos diferentes, pero uno y otro podían fijar previamente la precisión requerida, determinando el número de decimales del resultado porque el cálculo no suele ser exacto, introduciendo un cierto grado de error que se acotaba así. Hasta ahora habíamos tratado sumas y productos cuyos resultados son exactos.

El algoritmo de la división de manera general, si el numerador y denominador eran del mismo signo, se iba restando el denominador al numerador contando las veces, hasta que el signo del numerador cambiaba. Si tenían signos diferentes, se hacía lo mismo pero sumando el denominador al numerador. Para facilitar esta explicación se usan dos números positivos.

Se iniciaba la división restando el denominador al numerador, cada resta incrementaba en una unidad el acumulador cociente, repitiendo el ciclo hasta que el numerador pasaba a ser negativo cuando se producía el cambio de signo. En ese momento el valor del acumulador cociente era el resultado por exceso de la división, con un error máximo de una unidad, quedando en el numerador el resto sobrepasado de la división.

A continuación si se habían fijado cifras decimales, se desplazaba el valor sobrepasado del acumulador numerador un lugar a la izquierda, que era equivalente a multiplicarlo por diez. Con dicha cantidad negativa en el numerador, se continuaba el mismo proceso pero ahora sumando el denominador y restando una décima cada vez al cociente. Antes habíamos calculado el cociente real por exceso con un error máximo de una unidad, ahora se consigue una precisión de décimas en este paso, al aproximar restándolas.

Al volver a cambiar de signo, el numerador se tornaba positivo y se obtenía un cociente por defecto con un error máximo de una décima. Se repetía el proceso de desplazar o multiplicar por diez el acumulador numerador, se continuaba restando el denominador del numerador, con la exactitud de una centésima que incrementaba el cociente por cada resta. Y así se proseguía alternando procesos de suma y resta, hasta alcanzar la precisión de las cifras decimales requeridas. El cálculo se cerraba redondeando la última cifra decimal.

El valor inicial del numerador y del denominador junto con las cifras decimales establecidas, determinaban el número de veces que debía repetirse el proceso, en definitiva las sumas y restas a realizar, por ello los tiempos para efectuar una división eran variables; se solían considerar valores medios de tal manera que para d dígitos, con cinco decimales, hacían falta unos trece por d más uno ciclos de suma. La raíz cuadrada con un algoritmo diferente, realizaba un proceso parecido para aproximar el resultado mediante sumas y restas.

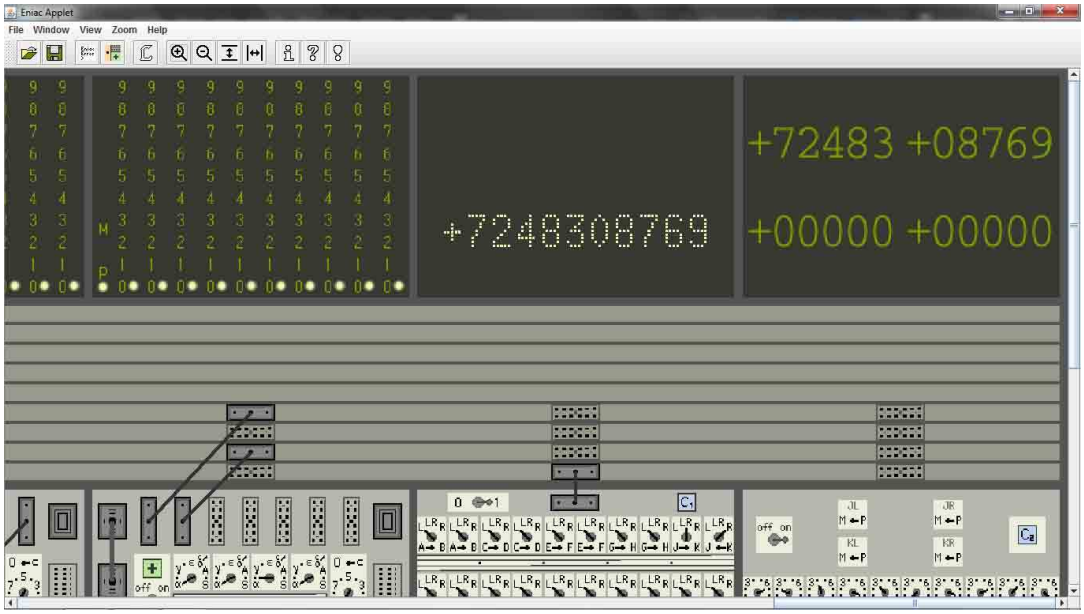


Imagen 37 – Cuadro de décadas y del transmisor de constantes

El algoritmo de la raíz cuadrada se basaba en la propiedad de la suma de los ene términos de una progresión aritmética, cuyo valor es el producto de ene por la suma del primer y el último término de la progresión, dividido por dos. Lo cual significa que si se hace una progresión aritmética de ene números impares consecutivos empezando por el uno, su suma vale ene al cuadrado. Se utilizaba esto en las calculadoras de la época para hallar las raíces cuadradas.

Si tenemos como radicando un número m cualquiera y queremos calcular su raíz cuadrada n, el algoritmo sencillamente era ir restándole números impares

consecutivos empezando por el uno, como se ve en la imagen 38 del simulador, hasta que cambie de signo o valga cero, el número de restas efectuadas n, era la raíz cuadrada del radicando m por exceso, con un error máximo de una unidad.

Imagen 38 – Inicio del cálculo de una raíz cuadrada

El algoritmo puede comprobarse fácilmente con radicandos pequeños como el dieciséis que es la suma de uno, tres, cinco y siete, cuatro impares luego su raíz cuadrada vale cuatro. Sin embargo este procedimiento de fuerza bruta que se ve en la imagen 39, era poco eficiente con números grandes. Con diez dígitos necesitaba unas setenta mil setecientas sumas y con números de veinte dígitos unas siete mil setenta … millones de sumas, a una velocidad de cálculo de cinco mil sumas por segundo que resultaba inoperativo. En estas explicaciones se va a utilizar el mismo ejemplo práctico, la raíz cuadrada de 7248308769.

Antes de comentar cómo solucionaron este problema, se deben exponer unas propiedades matemáticas, sin profundizar ni entrar en demostraciones, basadas en algunas peculiaridades. Si se analiza el valor de la suma de un grupo de diez impares, el grupo primero suma cien con los diez primeros impares, el grupo segundo trescientos con los diez impares siguientes, del once al veinte.

Aquí ya se observa una correlación entre el segundo impar que es el número tres y la suma del grupo segundo que son trescientos; el grupo tercero suma quinientos con sus correspondientes diez impares, y así sucesivamente. Por todo ello resulta que para el grupo n, sus diez impares suman *el impar n* por cien, o lo que es lo mismo suman *dos n menos uno* por cien.

Por otro lado, si la raíz del número m es n, o bien m es la suma de los n impares consecutivos, entonces m *por cien* es la suma de los n *por diez* impares consecutivos. Explicado con ejemplos, la raíz de uno es uno y la raíz de cien es diez; la raíz de cuatro son dos luego la raíz de cuatrocientos son veinte. O dicho de otra forma, si tres es la raíz de nueve, treinta es la raíz de novecientos.

Imagen 39 – Resultado de calcular una raíz cuadrada

También sucede que la raíz cuadrada de un número de dos dígitos tiene un dígito, pues se resuelve por exceso restándole un máximo de diez impares consecutivos y diez al cuadrado ya tiene tres dígitos. Por otro lado si se toma una cifra de diez dígitos, cuya raíz tiene cinco dígitos y se divide en cinco grupos de dos dígitos, teóricamente harían falta un máximo diez restas en cada uno de los cinco grupos para calcular la raíz parcial, más las operaciones para pasar de un grupo a otro, lo cual se aproxima bastante a la cifra media para resolver una raíz cuadrada de d cifras en el ENIAC que era de trece por d ciclos de suma.

Así se llega al mecanismo usado por el ENIAC para calcular las raíces cuadradas, que guardaba cierta relación con la aproximación de decimales en la división, pues eran series de restas y sumas, aproximando la raíz por exceso y defecto sucesivamente. Con números de diez cifras, resolvía por grupos de dos dígitos, comenzando por el de la izquierda, la década nueve y diez del *radicando* que estaba almacenado en *el acumulador numerador*.

Comenzaba con el impar uno en la década nueve del *acumulador denominador y raíz cuadrada*; e iba realizado *restas* de impares crecientes a partir de la década nueve, para comprenderlo gráficamente con ocho ceros por la derecha, *cada resta incrementaba* en uno la raíz hasta que el radicando

cambiaba de signo a negativo en menos de diez operaciones. Calculaba de este modo la raíz parcial de ese grupo *por exceso* con un error menor de la unidad.

Una vez resuelto este grupo uno, utilizando *el desplazador* multiplicaba por diez la raíz parcial anterior moviéndola una posición a la izquierda; luego situaba a partir de la séptima década el impar correspondiente a esa raíz, *dos n menos uno* porque era por exceso; para ello *el denominador* con el ultimo impar alcanzado lo desplazaba una posición a la derecha, lo dividía por diez y *sumaba* directamente nueve en la década siete, dejando en su posición al nuevo impar.

Con dicho emplazamiento en el denominador, que visto de forma gráfica tenía seis ceros por la derecha, ya se podía resolver el grupo dos con las décadas siete y ocho del radicando almacenado en *el numerador,* que no se movía en estas operaciones, y así desplazando *el denominador*, con los ceros a la derecha necesarios, conseguía ir calculando los grupos de dos dígitos del radicando.

72	Grupo 1
0	9
72	-9
1	17
9	***exceso***
Impares	Suma acu
1	1
3	4
5	9
7	16
9	25
11	36
13	49
15	64
17	**81**

48	Grupo 2
90	85
-852	23
179	171
-5	***defecto***
Impares	Suma acu
179	179
177	356
175	531
173	704
171	**875**

30	Grupo 3
850	852
2330	-1074
1.701	1703
2	***exceso***
Impares	Suma acu
1701	1.701
1703	**3.404**

87	Grupo 4
8.520	8513
-107.313	11.918
17.039	17.027
-7	***defecto***
Impares	Suma acu
17039	17.039
17037	34.076
17035	51.111
17033	68.144
17031	85.175
17029	102.204
17027	**119.231**

Raiz cuadrada

Radica./Nume.	
Impar/Denom.	
Sumas/restas	

69	Grupo 5
85.130	**85137**
1.191.869	**0**
170.261	170.273
7	***exceso***
Impares	Suma acu
170261	170.261
170263	340.524
170265	510.789
170267	681.056
170269	851.325
170271	1.021.596
170273	**1.191.869**

Imagen 40 – Cuadro ejemplo del cálculo de la raíz cuadrada de 72 48 30 87 69

Para facilitar la comprensión de este complejo proceso se añade un caso práctico de lo explicado con sus valores detallados. El cuadro de la imagen 40 contiene paso a paso los valores obtenidos mediante hoja de cálculo para el mismo número 72 48 30 87 69, cuya raíz cuadrada es 85137. Permite seguir esta explicación y además documentarlo sobre el ejemplo en cada punto.

Está dividido el cuadro de la imagen 40 en los cinco grupos de dos décadas, con los distintos valores que va tomando cada elemento en la resolución, tanto dentro de los grupos como en el paso entre grupos, usando el color azul para los cálculos parciales por exceso y el rojo para las raíces por defecto.

A destacar la evolución de la raíz cuadrada hasta alcanzar su valor 85137, debajo, el radicando con sus cambios de signo para marcar el final de operaciones en cada grupo, por exceso terminando en valor negativo y por defecto en positivo. Los impares usados dentro de cada grupo y el correspondiente al pasar de grupo. Más abajo hasta el final, se detallan las correspondientes series de impares usados con su valor acumulado.

Volviendo al proceso de cálculo de raíces del ENIAC, se tenía el grupo uno resuelto y se repetían las operaciones actuando sobre dos grupos del radicando. Antes se había resuelto por exceso, el radicando era negativo, por ello ahora se *sumaban impares decrecientes* hasta conseguir cambiar el signo del radicando a positivo, *decrementando* en uno la raíz por *cada suma* de impares, de tal modo que con menos de diez operaciones se obtenía la raíz parcial de ambos grupos *por defecto* con un error menor de la unidad.

Con los dos grupos resueltos se repetía el procedimiento de multiplicar por diez la raíz parcial usando el desplazador para llevarla una posición a la izquierda y se situaba a partir de quinta década al impar correspondiente, *dos n más uno* porque la raíz estaba calculada por defecto, para lo cual en *el denominador* desplazaba una posición a la derecha el último impar con el desplazador, que dividía por diez, además le *restaba* directamente nueve en la quinta década, esto dejaba colocado al impar para resolver otro grupo.

Se repetía el proceso como en el grupo inicial, calculaba la raíz *por exceso* con un error menor de la unidad, incrementando dicha raíz en cada *resta* de *impares crecientes,* hasta que el radicando cambiaba de signo a negativo. Y así sucesivamente alternado raíces parciales por exceso y por defecto hasta concluir los cinco grupos. Si se hubieran fijado decimales, se continuaría de manera análoga hacia la derecha, aunque para más de diez cifras con los segundos acumuladores. Por otro lado una vez concluidos los cálculos de la raíz había un proceso de redondeo para aproximar la última cifra, similar al de la división.

Una vez visto el algoritmo se destaca la eficiencia alcanzada con la resolución mediante cinco grupos de dos décadas, pues el resultado se lograba con treinta sumas y restas de impares, más las correspondientes operaciones para pasar cuatro veces de grupo. Cada paso de grupo suponía tres operaciones, el desplazamiento a la izquierda de la raíz parcial y el desplazamiento a la derecha del impar, sumando o restando nueve en la década adecuada, según

fuera por exceso o defecto. Frente a esta mejora, el cálculo inicial en fuerza empezando por el uno, costaba ochenta y cinco mil ciento treinta y siete sumas.

Este ejemplo de la imagen 40 y la explicación del algoritmo con los detalles del paso de grupo, se ha realizado por el autor del proyecto, ya que no se acaba de comprender. Se comenzó creando una hoja de cálculo sobre el número del ejemplo, cuya raíz cuadrada es exacta. Se han seguido los pasos y condiciones señaladas, haciendo pruebas o analizado con imaginación y sentido común hasta comprender los procesos y encontrar los valores adecuados para pasar de un grupo a otro, comprobando que se obtenía la raíz con su resto exactos.

Sobre todo se han estudiado los valores parciales de la raíz con su paso de grupo, la evolución del signo en el radicando, así como los impares a usar en cada grupo, con su correspondiente al cambiar de grupo, según fueran por exceso o defecto y el detalle de si son crecientes o decrecientes. Añadir que la raíz se guardaba en un acumulador que lógicamente será el del cociente.

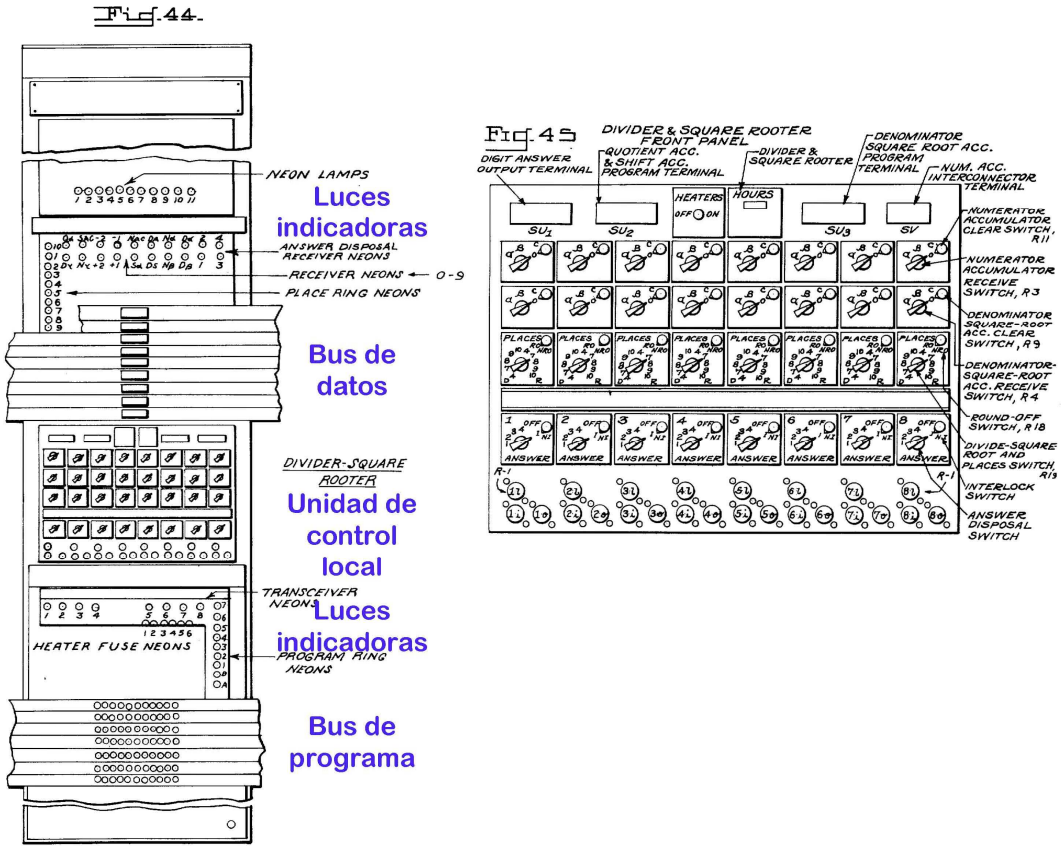


Imagen 41 – El divisor raíz cuadrada y la unidad de control

El patrón de ambos algoritmos, salvando los desplazamientos que no cambiaban, se basaba en restar al numerador un valor del denominador incrementado el cociente hasta que cambiaba de signo, para luego proceder a la

inversa, sumando el denominador disminuyendo el cociente hasta otro cambio de signo. El modelo lógico se podría cumplir cambiando solo el signo del denominador en cada cambio de signo del numerador, lo cual en el ENIAC se podría resolver simplemente conmutando los puertos de salida de S a A o viceversa, tanto para las actualizaciones de impares, como en los envíos de datos al numerador o las variaciones del cociente. Y con esto se puede dar por concluidos los dos algoritmos de división y raíz cuadrada.

Descripción del divisor/raíz cuadrada tenía una estructura similar al resto de elementos como muestra la imagen 41, con el bus de datos y el bus de programa para efectuar las conexiones necesarias de la programación. El modulo principal tenía una serie de indicadores luminosos para comprobar visualmente las posiciones establecidas en los interruptores. Entre ambos buses estaba la unidad de control donde se fijaban opciones mediante interruptores.

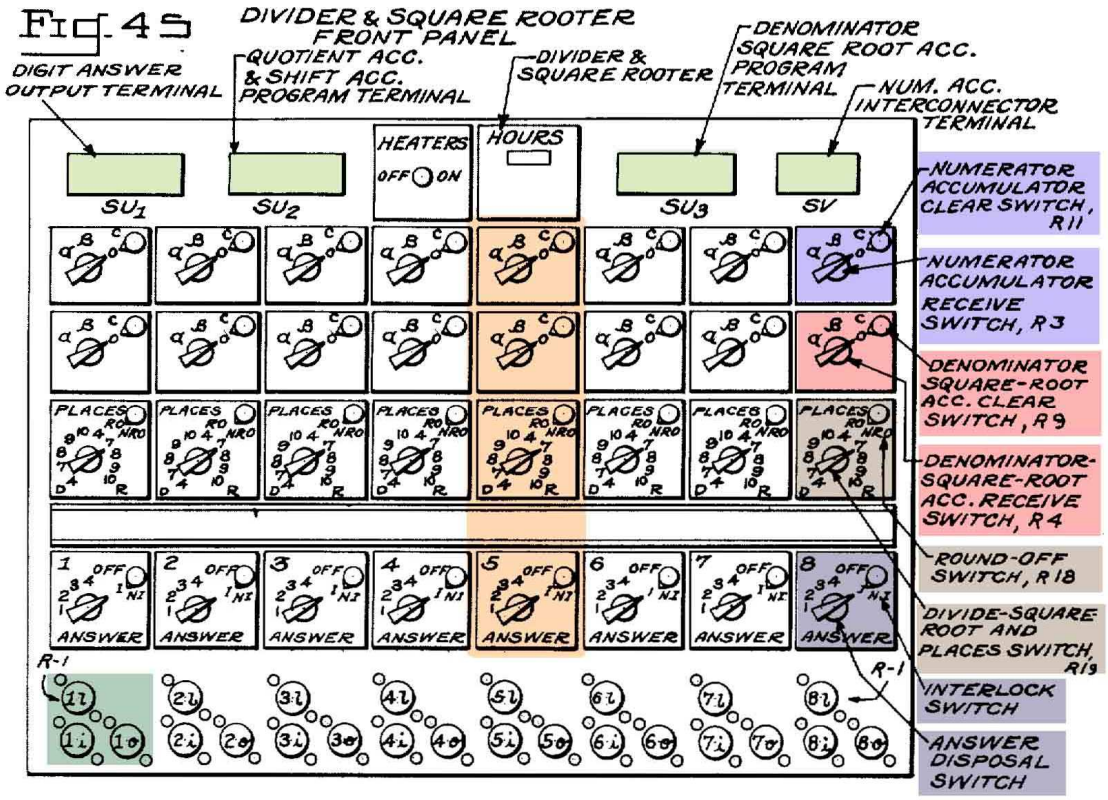


Imagen 42 – Detalle de la unidad de control del divisor raíz cuadrada

El detalle la unidad de control del divisor/raíz cuadrada con sus interruptores se muestra en la imagen 42, cuyo cometido se describe. La unidad de control de este módulo tenía en la parte superior las cuatro conexiones ya comentadas en las generalidades. Debajo podía establecer ocho configuraciones posibles con las correspondientes columnas de cuatro interruptores. En la parte

inferior había tres conexiones para cada configuración, con una conexión del grupo R-1 encima, más las tomas de entrada y salida de programa debajo.

Había cuatro interruptores dobles en cada columna, los dos primeros para el numerador y denominador, análogos, con su interruptor de borrado de los acumuladores al terminar situado arriba, y abajo a la izquierda estaba el interruptor de recepción para fijar la entrada alfa, beta o bien apagado. El tercer interruptor doble del módulo divisor/raíz cuadrada, tenía arriba el interruptor para encender el redondeo y abajo a la izquierda el interruptor para seleccionar división o raíz, con cuatro posiciones cada una, del siete al diez. El cuarto interruptor doble, arriba tenía el interruptor de bloqueo que permitía sincronizar la división o raíz con otras operaciones más rápidas y debajo a la izquierda estaba el de eliminación de respuesta con cuatro opciones, uno y dos para el cociente y la tres y cuatro para borrar la raíz.

Con este módulo se acaban las operaciones que podía realizar el ENIAC, donde hay una suma bastante rápida, con una duración media de doscientos microsegundos, otra la multiplicación más numerosa en los cálculos, con un tiempo aceptable de dos mil ochocientos microsegundos y finalmente una división y raíz cuadrada con una duración excesiva de veinte cuatro mil microsegundos. Esta disparidad de duración sin apenas mecanismos de control, hacía más complicado la programación paralela de varias operaciones, pues era un desperdicio de prestaciones programar al ritmo de las divisiones.

Por otro lado la falta de mecanismos de control creaba problemas añadidos, a resolver previamente por los programadores, pues por ejemplo el producto de dos números de seis y cinco cifras tenía entre once y nueve dígitos, lo cual obligaba a determinar con antelación si se realizaban los cálculos con veinte o diez dígitos. En la división y raíz cuadrada sucedía algo parecido al ir añadiendo precisión con cifras decimales, había que saber la modalidad de uso.

Y con esto se puede dar por concluidas las capacidades de cálculo del ENIAC, pasando a ver la memoria y las entradas o salidas que aumentaban sus prestaciones.

4.6. La memoria

La memoria del ENIAC era de dos tipos, una interna con los acumuladores y las tablas de funciones, y otra externa con las tarjetas perforadas. En la memoria interna estaban los veinte acumuladores de diez dígitos que ya se han visto en el apartado 4.5.1, además había tres tablas de funciones, cada una con dos módulos fijos y un tercero portátil. Por otra parte, la memoria externa consistía en las tarjetas perforadas que permitían al ENIAC en sus cálculos introducir

datos leyéndolas o imprimirlos perforando tarjetas, ayudado por el transmisor de constantes, la lectora y perforadora de tarjetas IBM.

4.6.1. La memoria interna

La tabla de funciones tenía una capacidad para almacenar ciento cuatro números de veinte dígitos con signo que solo se podían leer, pues su introducción era anterior y manual. Los datos de cada tabla tenían una estructura independiente, parecida a un vector de ciento cuatro elementos referenciados con un índice que iba desde el menos dos al ciento uno; cada número de veinte dígitos cuya estructura se muestra en la imagen 43, se podía usar completo con su signo o bien dividido en dos números con signo, de longitud *l* y otro de *veinte menos l*. Se almacenaban en la tabla de funciones datos de todo tipo como coeficientes, valores de variables o funciones, constantes y números trascendentes o cualquiera otro que fuera necesario.

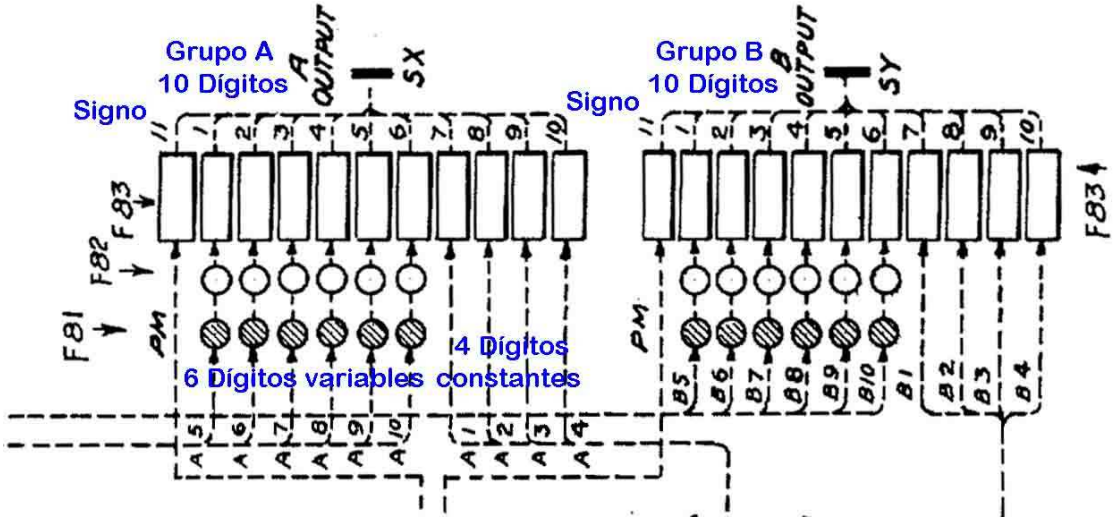


Imagen 43 – Tabla de funciones, detalle de un dato

Internamente cada constante almacenada se dividía en dos grupos A y B, de diez dígitos con signo; cada grupo tenía seis valores variables a establecer con los interruptores de la tabla de funciones portátil, que se pueden ver en la imagen 44 y cuatro dígitos constantes en el rango de la tabla, fijados por los interruptores de dígitos constantes del panel dos de la tabla de funciones. Las tablas de funciones usaban las dos formas de comunicación, estática con conexiones dedicadas y en serie o por pulsos, entre estas señalamos las dos terminales de salidas de función SX y SY de la imagen 43, que utilizaban cada una siete líneas para transmitir los grupo A y B, usando una para el signo y seis para los pulsos 1P, 2P, 2´P, 4P, 1´P y CPP.

Cada tabla de funciones constaba de tres elementos, dos módulos fijos que aparecen en la imagen 45 y uno móvil, la tabla de funciones portátil. Los fijos compuestos del panel uno y dos, tenían una distribución simular al resto del ENIAC, con los buses de datos y de programa para establecer las conexiones al programarlo y transmitir los datos requeridos, ambos paneles tenían unas luces para comprobar los interruptores fijados, y una unidad de control local compuesta por los dos paneles, que permitía programar el flujo de datos.

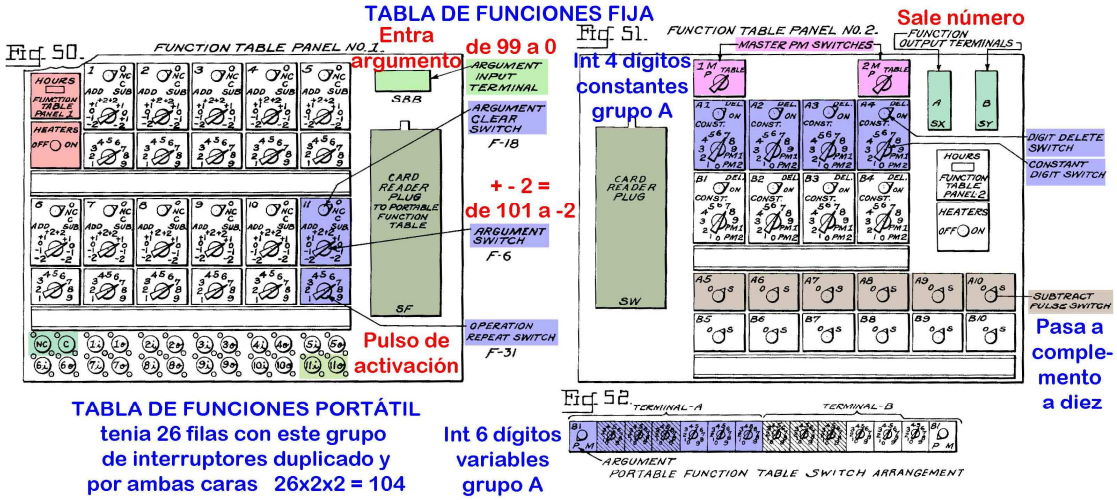


Imagen 44 – Unidad de control tabla de funciones

La unidad de control de la tabla de funciones se muestra en la imagen 44, cuyo panel uno en la parte lateral derecha tenía una toma de conexiones para la lectora de tarjetas a la tabla de funciones portátil con ciento cuatro líneas y arriba la terminal de entrada de argumento, argumento que era el índice para identificar el dato. Esta entrada fijaba parcialmente el índice con las décadas decenas y unidades de un acumulador auxiliar, pero solo entre cero y noventa y nueve. En la parte inferior había dos tomas puerto de salida *NC/C* para mantener o borrar el argumento al final de la operación y once pares de tomas entrada salida para conectar cada configuración al bus de programa.

Los interruptores del panel uno permitían once configuraciones de lectura de datos. Cada configuración tenía dos interruptores, el superior doble con un interruptor de borrado de argumento y un interruptor de argumento que añadía desde menos dos a más dos al argumento parcial anterior, de este modo, ver imagen 44, se completaba y podían referenciarse los ciento cuatro índices con los argumentos entre menos dos y ciento uno.

Además este mismo interruptor de argumento señalaba si el valor se sumaba con *ADD* o se restaba *SUB* que realmente lo enviaba en complemento a nueve, necesitando luego un pulso 1´P para pasarlo a complemento a diez. Debajo el interruptor de repeticiones fijaba las veces que se hacía la consulta.

La unidad de control del panel dos, tenía en ambos lados conexiones para la lectora de tarjetas, el conector de la izquierda especificaba los cuatro dígitos constantes y a la derecha estaban los terminales de salida de función grupo A y B del número almacenado. En la parte central superior tenía los interruptores de signo PM maestro uno y dos, cada uno con tres posiciones para el signo, P y M si era constante o bien en tabla, si era variable.

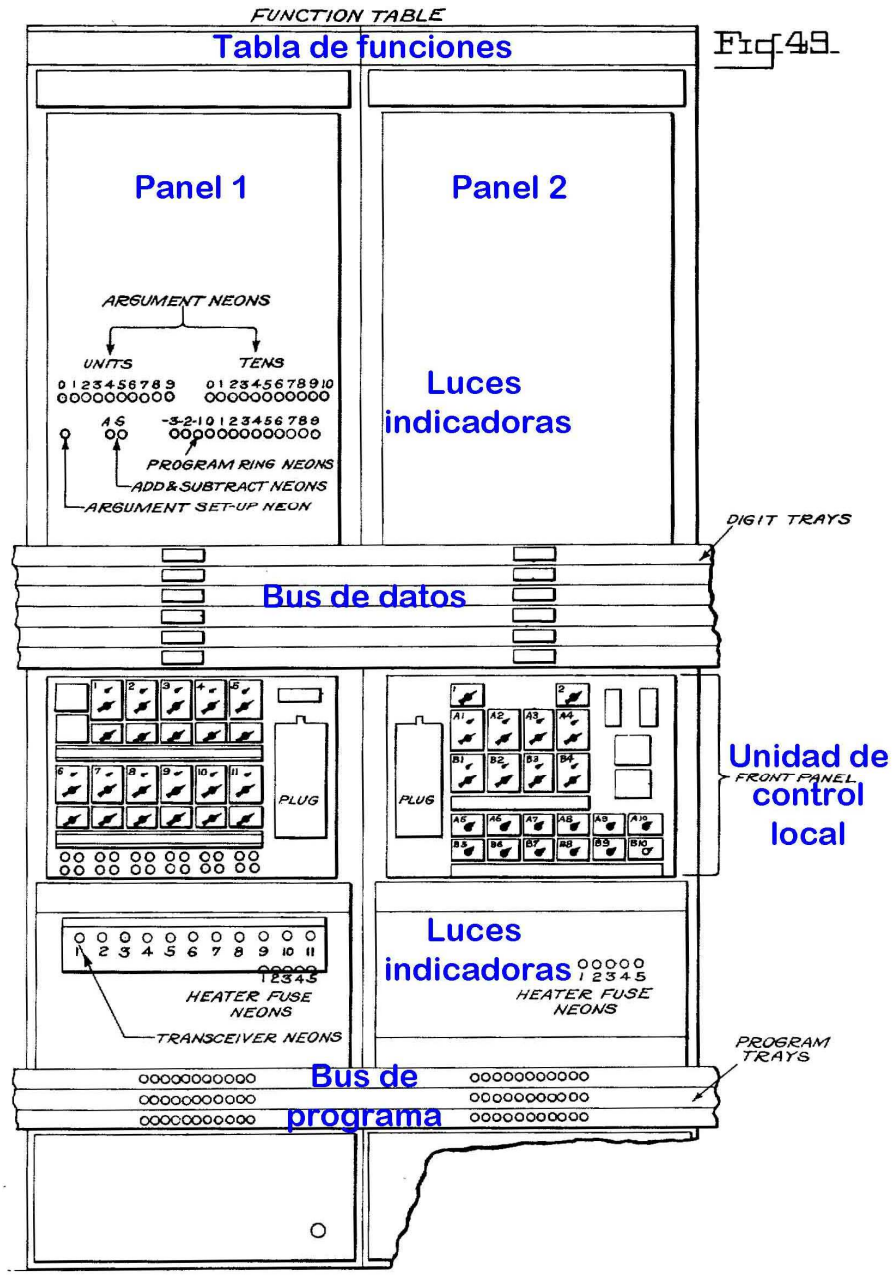


Imagen 45 – La Tabla de funciones, elementos fijos

Debajo tenía dos series de cuatro interruptores dobles de dígitos constantes para el grupo A y B. Cada uno tenía un interruptor en la parte superior para

activar el borrado que impedía transmitir pulsos de dígitos y otro interruptor debajo para fijar uno de los cuatro dígitos constantes de cada número, se colocaba su valor y el signo que era el del interruptor de signo PM seleccionado.

En la parte inferior había dos filas de seis interruptoras de pulso de resta para los grupos A y B, que se usaban para completar la transmisión restando *SUB* del panel uno explicada antes, si se activaba permitía a los seis dígitos variables recoger el pulso 1´P que necesitaban para pasar a complemento a diez, normalmente solo se establecía un grupo en esta posición.

La tabla de funciones portátil era un panel con veintiséis filas de interruptores por las dos caras, donde cada fila tenía dos conjuntos de catorce interruptores como el mostrado en la imagen 44. En la fig 52 de la patente se ven los catorce interruptores, siete interruptores para cada grupo A y B, con un interruptor para el signo y seis para los seis dígitos variables.

En conjunto las tres tablas de funciones del ENIAC tenían capacidad para almacenar trescientos doce números de veinte dígitos con su signo, que se podían utilizar completos o partidos en dos. El tiempo de lectura de datos como ya se indicó en el apartado 4.3 era de cuatro ciclos de sumas más un ciclo por cada número leído. Las tablas de funciones tenían conexiones estáticas pero enviaban sus números con conexiones de pulsos por las salidas del panel dos.

4.6.2. La memoria externa

La memoria externa se basaba en el uso de tarjetas perforadas tanto para leer datos como para guardar resultados. Las tarjetas perforadas tenían ochenta columnas de doce filas, con lo cual podían almacenar ocho números de diez dígitos que se solían etiquetar de la A a la H, como se aprecia en la imagen 46. Cada columna tenía doce filas usándose diez, de la cero a nueve para almacenar el dígito que se perforaba. El signo figuraba en la fila once, se solía considerar el de la cifra más significativa, a la izquierda de cada número. Los números negativos se convertían a complemento a nueve durante la lectura para pasarlos a complemento a diez en la transmisión, añadiendo el pulso 1´P necesario.

Las ochenta columnas también se podían dividir en dieciséis números de cinco dígitos, como se observa en la imagen 46 y con el transmisor de constantes se fijaba el modo de lectura de datos. La tarjetas perforadas daban una gran capacidad al ENIAC, al poder guardar y leer los cálculos, también le daba mucha flexibilidad, pues con cálculos paralelos difíciles de coordinar permitía comenzar calculando operaciones e imprimir los resultados, para continuar luego leyendo los resultados de las tarjetas, posibilitando concluir los cálculos completos. Como dato en los primeros cálculos solicitaron un millón de tarjetas.

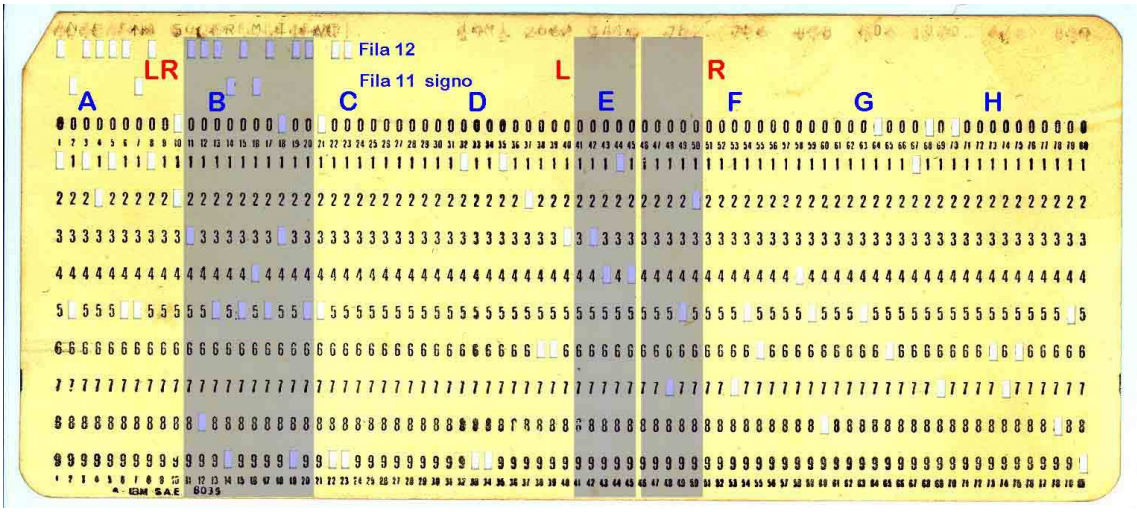


Imagen 46 – Tarjeta perforada de 80 columnas

4.7. La entrada/salida

Los dispositivos de entrada se agrupaban en torno al transmisor de constantes al que se conectaba la lectora de tarjetas. Como mecanismo de salida estaba la perforadora de tarjetas, que entonces consideraban una impresora y se conectaba a través los tres módulos de impresora del ENIAC, recibiendo los números a imprimir de los acumuladores que los almacenaban. Los dispositivos finales, la lectora y la perforadora realmente eran propiedad de IBM que los alquilaba mensualmente por unos ciento sesenta dólares de la época.

También se pueden considerar un elemento de salida los cuadros de luces con las décadas de los acumuladores, al concluir los cálculos mostraban los resultados. Estas luces se usaban igualmente como medio de control durante el funcionamiento, pues al desplazarse por filas permitían detectar directamente cuando fallaban, señalando al contador de décadas con posibles problemas.

4.7.1. El transmisor de constantes

El transmisor de constantes era un elemento de enlace que servía para enviar datos al ENIAC desde la lectora de tarjetas o bien podía cargar sus dos números de diez dígitos con signo almacenados. Se componía de tres módulos, el panel uno, el dos y un módulo a la derecha con conexiones. Los paneles uno y dos tenían una disposición similar al resto, con los buses de datos y de programa, los indicadores luminosos para comprobar los elementos y una unidad de control local que integraba ambos paneles. Por otro lado el tercer

módulo con circuitos internos, en el exterior tenía tres conexiones, una mayor en el centro, para la lectora de tarjetas y dos a los lados para conexiones propias.

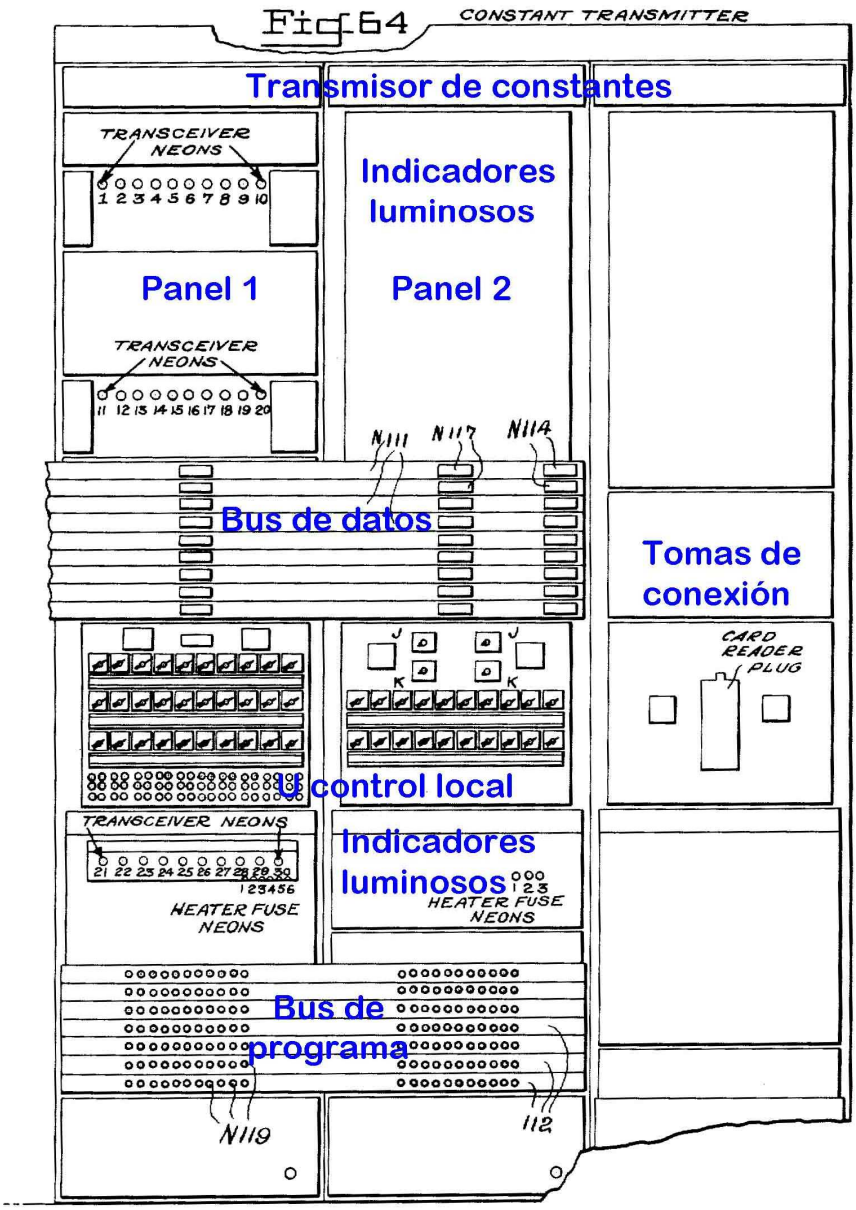


Imagen 47 – El transmisor de constantes

El transmisor de constantes solo tenía un puerto de salida en el panel uno, por ello podía enviar un dato por cada ciclo de suma. En la imagen 48 que lleva en amarillo las partes reproducidas en el simulador eniac.jar, se aprecia dicha salida. El transmisor de constantes almacenaba los números de las tarjetas perforadas mediante un grupo de trescientos veinte interruptores de relés que se fijaban según dichos valores. Se basaba en que la lectora de tarjetas al detectar una perforación en un lugar, enviaba una señal eléctrica al

correspondiente relé del transmisor de constantes que le permitía almacenar la información de la tarjeta en el ENIAC para su posterior envío.

La unidad de control local abarcaba los paneles uno y dos mostrados en la imagen 48. En el panel uno había tres filas de diez interruptores de selección, ocho para transmitir los números de las tarjetas perforadas, etiquetados de la A a la H tal como se vio en la imagen 46 al explicar la tarjeta, y otros dos etiquetados K y L para los dos suyos. Los números con signo de diez dígitos, en general así como en las tarjetas, se podían usar completos con la opción *LR* o bien considerarlos divididos en dos cifras de cinco dígitos con signo, empleando la etiqueta *L* para los de la izquierda y *R* para los de la derecha.

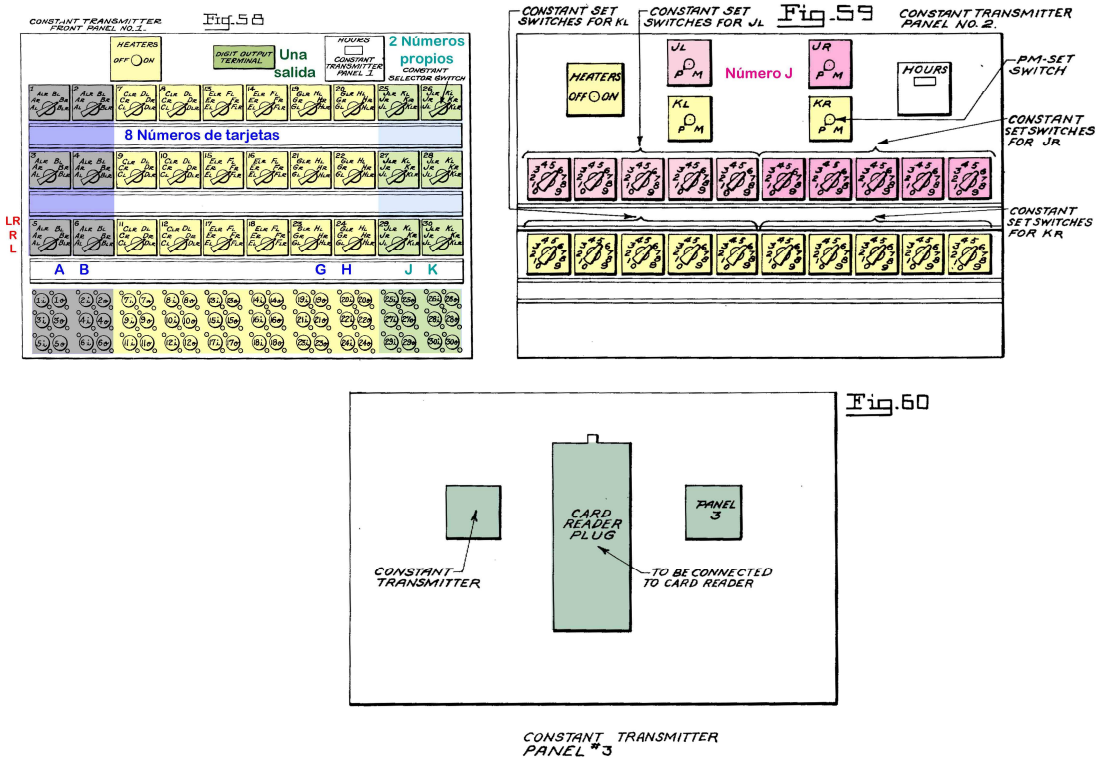


Imagen 48 – El control local del transmisor de constantes

Los treinta interruptores como se ve en la imagen 48, se compartían por las parejas de números de diez dígitos, comenzando por la A, B. Estaban duplicados formando grupos de dos en tres filas, disponiendo por tanto cada pareja de seis interruptores iguales para poder utilizarlos de varias formas independientes. Cada interruptor tenía seis posiciones, tres para cada miembro de la pareja fijando las opciones *L, R* o *LR* en su número. A cada interruptor le correspondían, situadas en la parte inferior, sus dos tomas, entrada y salida, para conectarse al bus de programa, permitiendo recibir una activación de envío de datos o bien enviar un pulso al finalizar, para activar la siguiente etapa.

En el ENIAC, el ajuste del tamaño del número a transmitir, en cada grupo de seis interruptores de las parejas, tenía que ser único, para decirlo de manera fácil todos de cinco o bien de diez dígitos. Sin embargo en el simulador eniac.jar se permite enviar los números del transmisor con cinco y diez dígitos.

El panel dos de la unidad de control servía para fijar los números del transmisor de constantes, denominados J y K de diez dígitos con signo. Tenía dos filas de interruptores arriba, cada una con dos interruptores para fijar el signo de la parte izquierda o la derecha del número correspondiente, con las opciones P o M ya explicadas. Debajo tenía dos filas de diez interruptores para fijar cada cifra del cero al nueve, en los respectivos números.

4.7.2. La lectora y la perforadora de tarjetas

La lectora de tarjetas IBM era un dispositivo mecánico, ver imagen 50, que podía leer el contenido de cada tarjeta en las ochenta columnas, comprobando en cada posición la existencia de la perforación, que se traducía en una señal eléctrica la cual activaba los correspondientes interruptores de relés del transmisor de constantes, de esta manera la información de la tarjeta pasaba al transmisor de constantes, pudiendo enviarse luego al resto del ENIAC.

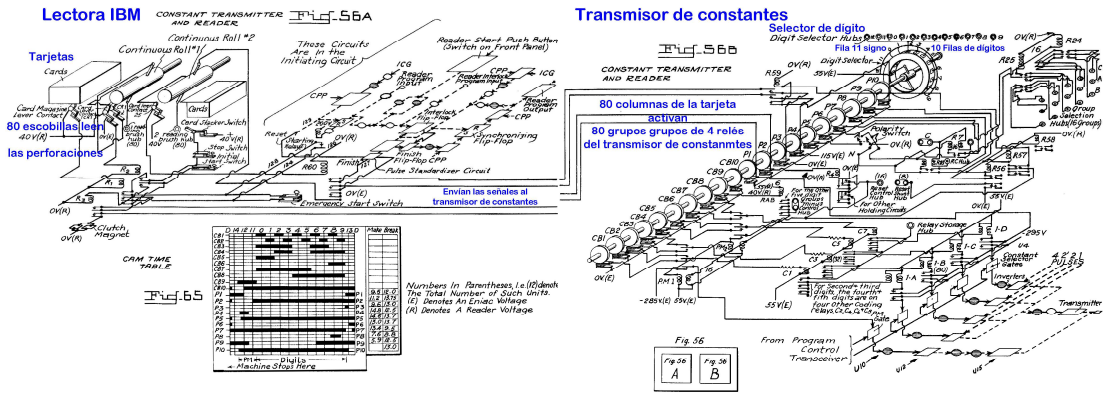


Imagen 49 – Lectura de tarjetas y su almacenamiento en los relés

La lectora de tarjetas tenía una bandeja cargadora a la izquierda de la imagen 49, donde colocaban las tarjetas en montones que se leían a una velocidad media de ciento veinticinco tarjetas por minuto, aproximadamente medio segundo por tarjeta. El proceso arrancaba desde la bandeja, se arrastraba con unos rodillos la primera tarjeta hacia la derecha, se leía por filas de arriba abajo al desplazarse, para lo cual disponía de ochenta escobillas metálicas en contacto con la tarjeta, una por cada columna.

Al pasar por una perforación su escobilla cerraba el circuito en las levas de contacto, produciendo una corriente eléctrica que la lectora IBM convertía en

las señales de control de cuatro relés, las cuales se enviaban al transmisor de constantes activando con pulsos los correspondientes cuatro relés que controlaban las puertas de paso de los trenes de pulsos 1P, 2P, 2´P y 4P de la unidad de ciclos y así se almacenaba el dígito. Todo este proceso se producía simultáneamente en las ochenta columnas a ambos lados.

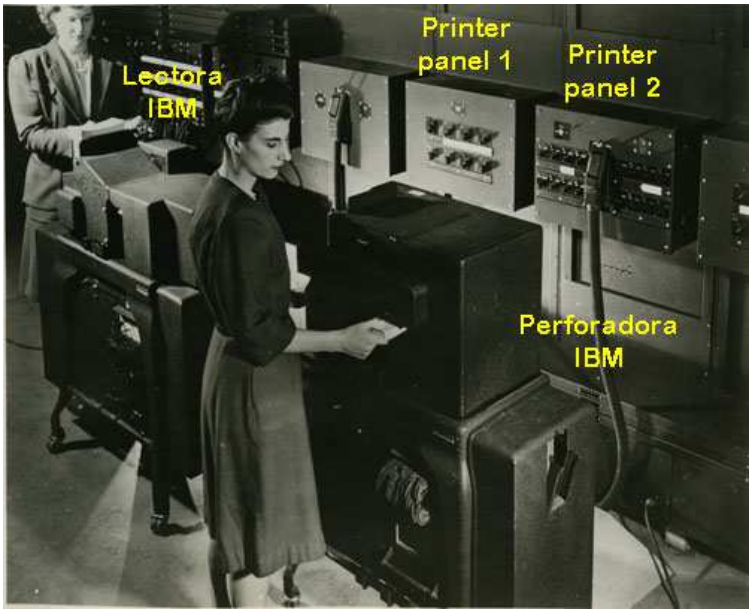


Imagen 50 – JJ. Bartik y F. Bilas con la lectora y perforadora IBM

El resultado era que el transmisor de constantes recibía por cada columna las cuatro señales que activan adecuadamente sus cuatro relés permitiendo el paso de los pulsos necesarios de la unidad de ciclos, que daban como resultado el almacenamiento de los números de la tarjeta perforada. El transmisor de constantes para este proceso empleaba trescientos veinte relés en ochenta grupos de cuatro.

La perforadora de tarjetas IBM era el complemento de la lectora, ambas se muestran en la imagen 50, servía para imprimir los resultados de los cálculos y se controlaba desde los tres últimos módulos del ENIAC, los paneles de impresora uno a tres, que se muestran en la imagen 51.

La memoria junto con los mecanismos de entrada y salida aumentaban las prestaciones del ENIAC, ayudando a paliar sus problemas de falta de control, pues como última solución se podía calcular a la mayor rapidez e imprimir los resultados, siendo sencillo luego terminar los cálculos a partir de las tarjetas. El ENIAC tenía algunos procesos que se podían hacer a mano, pero los tiempos subían a segundos o minutos en el mejor de los casos, por ello resultaba complicado manejarlo manualmente sin perder prestaciones.

En la memoria con un poco de perspectiva se observa un cierto escalonamiento, en un primer nivel estaban los acumuladores que servían de registros rápidos, en el segundo nivel disponía de las tres tablas de funciones para cargar valores, sin comunicación entre ellas ni escritura, que eran introducidos a mano previamente, tenían un tiempo de acceso por lectura aceptable y en el tercer nivel se situaba la información de las tarjetas, memoria mucho más amplia pero bastante más lenta.

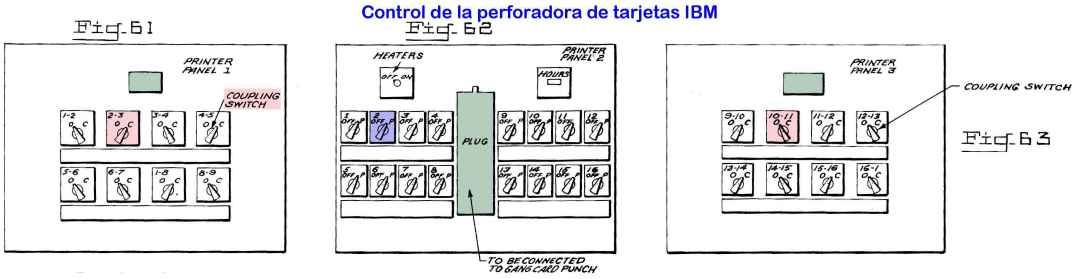


Imagen 51 – Paneles de control de la perforadora IBM

El ENIAC no tenía teclado, pero se podían, metafóricamente hablando, escribir los datos en los acumuladores o las instrucciones, conectando sus cables y fijando los interruptores pertinentes. Tampoco tenía programa almacenado pero disponían en papel de las rutinas para los cálculos, como el mostrado en la imagen 52, utilizándose posteriormente el programa como esquema para realizar directamente las conexiones y fijar los interruptores, realmente disponía de un archivo de programas pero era externo porque había que realizar su montaje previo, dado que se programaba manipulando la propia máquina.

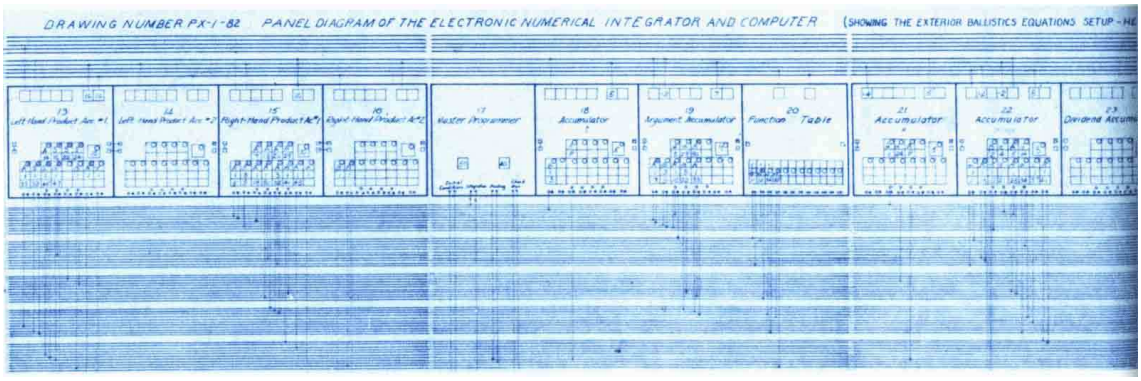


Imagen 52 – Fragmento de programa del ENIAC

Así podemos dar por concluida la descripción del ENIAC y sus peculiaridades, se ha buscado dar una explicación apoyada en las imágenes para captar con profundidad su funcionamiento sin perderse en la complejidad de sus vetustos circuitos y mecanismos.

5. Programación del ENIAC

El ENIAC fue retirado de servicio hace casi sesenta años, se podría considerar desaparecido pero en los últimos tiempos se ha resucitado de algún modo, al volver a mostrarse interés por investigar y conocerlo. Se destacan dos proyectos, el *ENIAC-on-a-Chip* para construir una réplica del computador, con su lógica y funcionamiento controlados mediante un chip, desarrollado por la Universidad de Pennsylvania y un simulador realizado por Till Zoppke de la Universidad libre de Berlín (Zoppke, 2013), solo es un *applet* de java, pero constituye la herramienta usada para desarrollar los programas del ENIAC.

5.1. El simulador eniac.jar

Esta aplicación denominada eniac.jar tiene varios modelos disponibles en internet con el mismo nombre, de tal manera que se comenzó buscando, probando y seleccionando la versión eniac.jar de cuatrocientos treinta y siete kilobytes, para identificarla. El motivo principal fue que solo esta funcionaba correctamente en el ordenador propio. La aplicación es un simulador pero hay que tener presente que no actúa exactamente igual que el ENIAC, además algunas cuestiones físicas o reales no se pueden simular. Por ello no se debe perder de vista la idea de que un simulador es una aproximación al objeto real que nos sirve para comprenderlo, pero no es el objeto.

Asimismo la simulación es parcial, abarca veinte de los cuarenta y cinco módulos. No se incluyen partes importantes del ENIAC como se ve en la imagen 53, que lleva señalados en rojo los componentes no simulados. Solo están desarrollados la unidad de inicio, de ciclos, dieciséis acumuladores y dos paneles del transmisor de constantes. El resultado es un ENIAC básico, sin controles, muy menguado en sus prestaciones, que por señalar una cifra se podría estimar con un veinticinco por ciento de sus capacidades iniciales.

La programación planteaba varios frentes que la hacían difícil. El principal problema era el desconocimiento de estos complicados modos de programar (Zoppke, 2006), que por ello desaparecieron directamente con el ENIAC. La falta de conocimiento se agravaba con la falta de aplicaciones para comprender cómo se realizaba o se podrían desarrollar programas nuevos.

Se han encontrado cuatro programas que funcionan sobre este simulador, tres forman parte del mismo, con el ejemplo sencillo de un producto y una suma, el algoritmo de Euclides que calcula el máximo común divisor de dos números y el cálculo de una sucesión de Fibonacci (Zoppke, 2004), usando

veinte dígitos. Por otro lado, se ha localizado otra aplicación de la Universidad de Potsdam (Schapranow, 2006), que desarrolla el módulo para calcular el resto de la división de dos números.

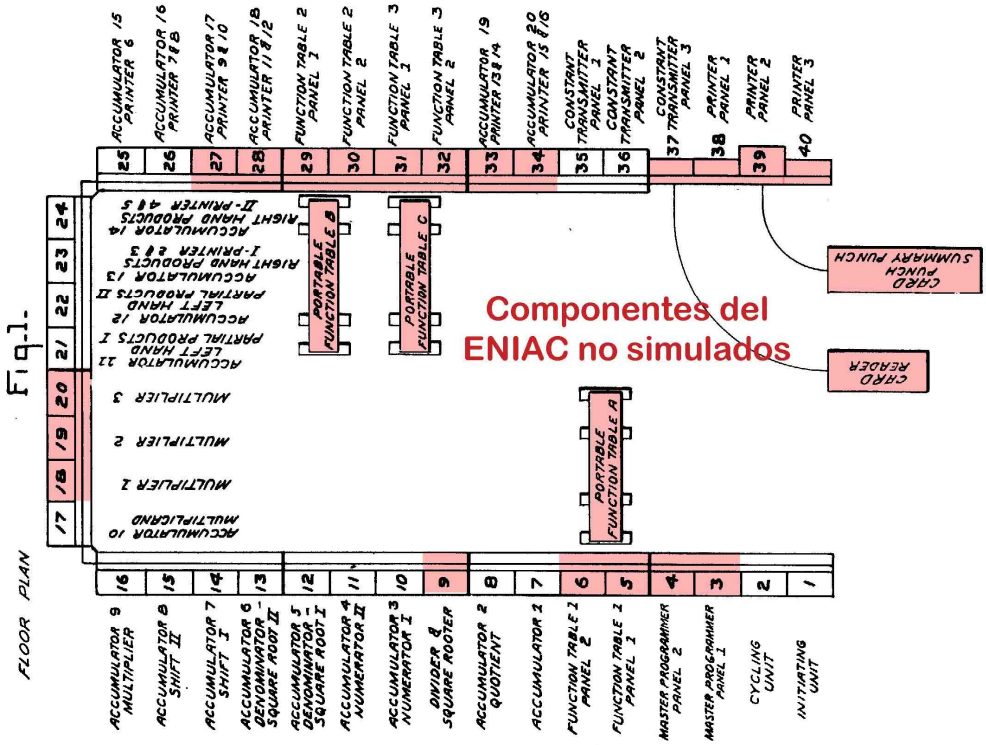


Imagen 53 – Componentes simulados del ENIAC

A partir de este punto se estuvo más de un mes sencillamente investigando sobre el ENIAC sin otro objetivo que conocerlo y aprender a programarlo.

5.2. La programación

La programación del ENIAC, se planteó de forma general desarrollando procedimientos o rutinas para efectuar ciertas tareas de manera modular, permitiendo intercambiar fragmentos de una aplicación a otra. Los principios usados para desarrollar los programas fueron la sencillez, limpieza, uniformidad y sentido práctico. Se comenzó analizando los medios disponibles en el simulador, empezando por la unidad de inicio y de ciclos que se muestran en la imagen 54, son dos módulos necesarios para la puesta en marcha, pero realmente no resuelven los problemas.

En la imagen del simulador se ha señalado de manera esquemática los cometidos que realiza cada elemento, pues el funcionamiento de estas unidades

ya fue explicado en el apartado 4.3. Se señala que en la unidad de ciclos han añadido un regulador de velocidad que permite modificarla en el simulador, así como un contador de ciclos de suma, que resulta útil para seguir operaciones o medir la velocidad real del simulador, también han mejorado bastante el osciloscopio original. Hay dos pulsadores importantes en la unidad de inicio porque los programas están preparados para manejarse por cualquier usuario solo con pulsar el botón de arrancar, "*go!*" y una vez terminado el cálculo, se puede borrar con el botón "*clear*" y volver a arrancar cuantas veces se desee.

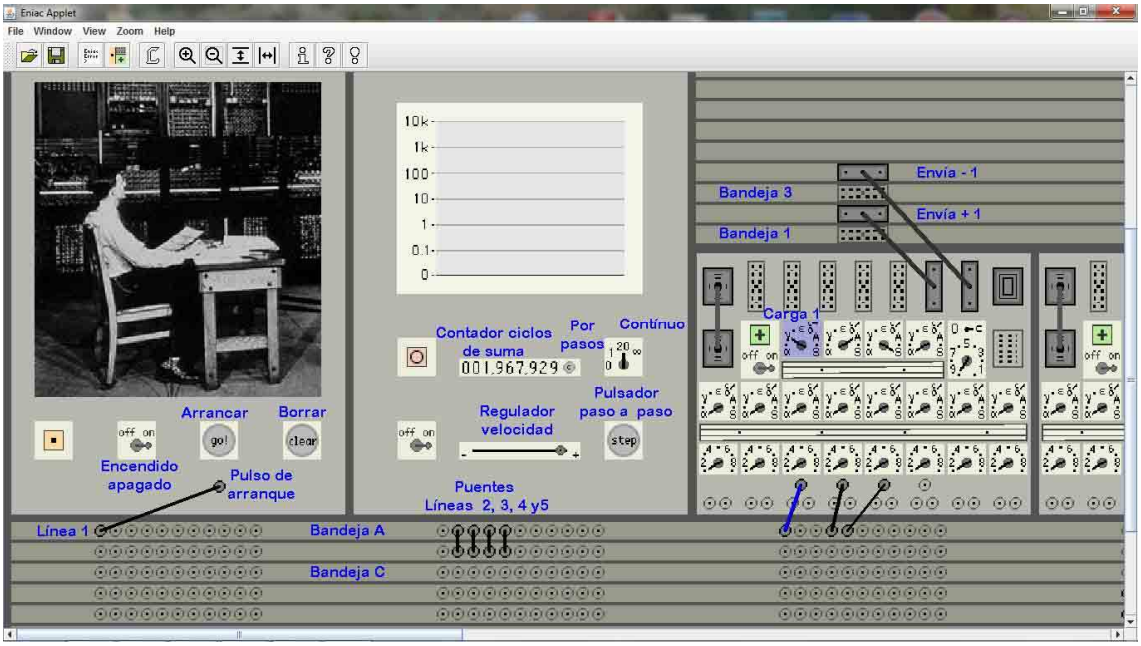


Imagen 54 – La unidad de inicio, de ciclos y un acumulador

Al final del ENIAC estaba el transmisor de constantes que ahora solo permite almacenar dos números de diez dígitos con signo y enviarlos en cualquier momento, pues sin la lectora de tarjetas no se tiene memoria externa. Se puede ver en la imagen 55, es similar al descrito en el apartado 4.7.1, aunque en el simulador le han añadido dos cuadros luminosos que facilitan la información de los números almacenados o al transmitirse, como en este caso enviando un cuatro. También se puede observar el único puerto de conexión de datos del transmisor de constantes, que solo le permite enviar uno por ciclo.

Prácticamente quedaban los dieciséis acumuladores como peones para crear programas, pero solo podían sumar o restar, lo cual hace muy lentos los procesos de multiplicar, dividir o hallar raíces cuadradas, verdaderos motores del cálculo en el ENIAC original. Repasando los buses de datos y de programa, el simulador contaba con cuatro bandejas de datos como se ve en la imagen 54, etiquetadas de la uno a la cuatro, con una conexión por elemento y debajo cinco bandejas de programa, de la A a la E, cada una con once tomas individuales de programa, numeradas de la uno a la once, a partir de la izquierda.

Se comenzó aprendiendo a arrancar el ENIAC y enviar recibir números de un acumulador a otro, siguiendo la dinámica de mandar un pulso de arranque, realizar las tareas y lanzar otro pulso al finalizar. Así se fueron creando unas normas y definiciones particulares para estructurar las tareas. Dos de estos conceptos son la fase y la etapa. El ENIAC arrancaba enviando un pulso, que desencadenaba unas acciones en los elementos conectados y al acabar, uno de ellos enviaba otro pulso para arrancar la siguiente serie de acciones.

El conjunto de operaciones desencadenadas en cada pulso se considera una etapa, que puede activar la siguiente con otro pulso o bien no hacer nada en la etapa final. Así un programa se convertía en una secuencia de etapas delimitadas por los pulsos de arranque. Otra cuestión buscada fue el orden, los pulsos se iniciaban de izquierda a derecha y de arriba abajo, pues resultaba muy complicado seguir un programa que lanzaba el pulso arranque en la línea nueve.

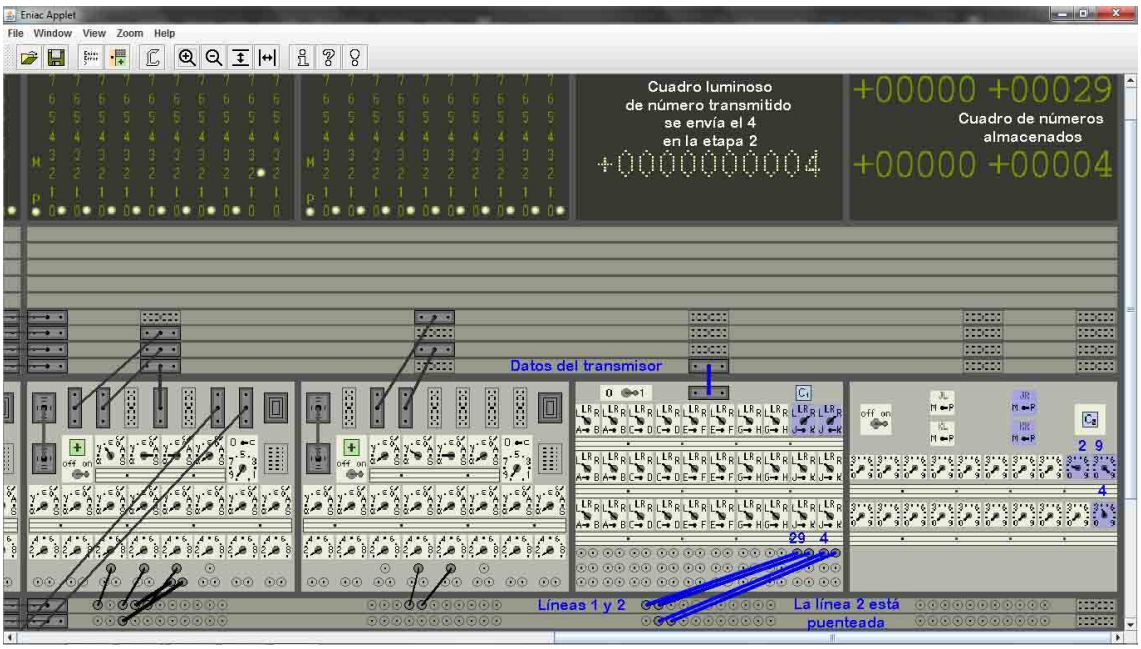


Imagen 55 – El transmisor de constantes

La estructura general de los programas es la misma en todos porque se ha previsto utilizarse solo con los botones de arranque y borrado. Se parte de una *situación inicial* con todos los acumuladores a cero y se tienen *tres fases*, de *carga de datos,* de *ejecución cíclica del algoritmo* y de *restauración* del último estado de la máquina. Luego, dentro de cada fase estaban las etapas necesarias para desarrollar las acciones programadas, buscando paralelizar al máximo para reducir el número de etapas, pues el simulador es lento y tiene pocos recursos.

La primera fase, *carga de datos*, requería como mínimo tantas etapas como datos se usan, pues el transmisor de constantes con cada pulso solo transmitía uno. En la imagen 55 se aprecia como para la división se transmite el dividendo

veintinueve y el divisor cuatro. Otro aspecto importante son los puentes, ver la imagen 58, en el bus de programa el transmisor de constantes necesita dos conexiones a la línea dos, una para lanzar el pulso de activación por esa línea concluida la etapa uno y además necesita otra conexión que reciba ese mismo pulso para arrancar su propia siguiente acción, enviar el cuatro en la segunda etapa, que se soluciona puenteando ambas líneas en la unidad de ciclos.

La segunda fase de *ejecución cíclica del algoritmo,* es el programa propiamente, que se repite hasta alcanzar la condición. La condición se alcanza siempre sobrepasándola. La bifurcación condicional como se vio en la discriminación de magnitud del apartado 4.5.1 sólo comprobaba la condición menor que, lo cual obligaba a realizar una iteración más, dicho de forma fácil se ejecuta un ciclo de más. Esto tenía varias soluciones y se escogió añadir una tercera fase de *restauración* del último estado de la máquina antes del sobrepaso, por ser mejor y más didáctica, permite aprovechar el resto obtenido y observar el estado de los acumuladores cuando se alcanza la condición.

Los programas desarrollados tuvieron una evolución, partiendo de la instrucción disponible a desarrollar, la bifurcación condicional o "*if*", se realizó una implementación con dos caminos y enseguida se vio la necesidad de disponer siempre del uno, para alimentar un contador sumando o restando, pues casi todas las operaciones lo usaban. Por ello el primer acumulador de los programas se configura como se ve en la imagen 54 para cargar el uno y sumarlo o restarlo después. Cuando se superó esta parte de programación básica se volvió a analizar el ENIAC original para plantear los programas a realizar.

De entrada se vio conveniente desarrollar los programas, se podrían denominar de propósito general, que recuperaban las operaciones básicas del ENIAC, multiplicación, división y raíz cuadrada; aunque ahora resueltas mediante sumas o restas. En la división y raíz cuadrada no era factible calcular decimales, pues no se contaba con los acumuladores de desplazamiento ni el componente que los gobernaba. Para desarrollar los programas se marcó una condición importante, el coste del cálculo debía ser lineal con la talla, descartando costes exponenciales, que consumen mucho tiempo y recursos.

Por otra parte al estudiar la raíz cuadrada, se observó que aprovechaba la propiedad matemática de la suma de los ene términos de una progresión aritmética, cuyo valor es el producto de ene por la suma del primer y el último término de la progresión, dividido por dos. Propiedad utilizada para calcular las raíces cuadradas de un número, sumando impares consecutivos a partir del uno, como se ha visto en el apartado 4.5.3, hasta igualar o superar el número.

Esto llevó a profundizar con la suma de los términos de una progresión aritmética, deduciendo la manera de elevar un número al cubo con un coste lineal. Estudiando la fórmula se calculó que si el primer término de la progresión aritmética era el mismo número y se incrementaba cada vez con su

doble, la suma daba ese número al cubo, pero con una talla ene. También se planteó por el tutor desarrollar un programa similar a la prueba del ENIAC en mayo de mil novecientos cuarenta y cuatro, para ejecutar sumas continuadas entre dos acumuladores y medir su velocidad de cálculo, así como otro programa para calcular la suma los ene primeros números.

En la parte teórica se estudiaron las fórmulas físicas del movimiento de un proyectil en el espacio, con y sin rozamiento, cálculos que en principio eran el objetivo del ENIAC. Por otro lado, según se fueron probando y superando los programas individuales, se vio la posibilidad de crear una aplicación que integrara varias, para calcular una trayectoria sin rozamiento, que se acabó consiguiendo apurando los medios. El problema de la trayectoria se resumía en introducir cuatro valores y ejecutar con ellos seis multiplicaciones y una división de un resultado parcial, empleando solo los dieciséis acumuladores.

El desarrollo de los programas para el ENIAC tiene una finalidad didáctica de mostrar el proceso seguido y analizar los acumuladores utilizados con su estado, sobre todo en la iteración que se alcanza la condición. Los programas están pensados para usuarios que no conocen el ENIAC, y sus operaciones se reducen a arrancar el simulador con el botón *go!* y una vez finalizado el cálculo se puede repetir, poniendo a cero con el botón *clear* y volviendo a arrancar. Conviene señalar que al finalizar un cálculo los acumuladores no se pueden volver a usar sin ponerlos a cero y cuando se borran pierden los resultados, en el ENIAC original los cálculos se podían almacenar imprimiéndolos en tarjetas.

Todo ello ha llevado a desarrollar este manejo del simulador ENIAC por usuarios inexpertos, denominado particularmente en modo semiautomático, el cual permite además cambiar los valores del transmisor de constantes, modificando sus interruptores, para calcular otros resultados. Y de todos modos siempre se puede usar en modo manual, como se hacía en el ENIAC original, colocando los valores de los acumuladores, cableando o manipulando los interruptores directamente, pero eso ya requiere saber programarlo.

En el ENIAC original el multiplicador y divisor contaban con mecanismos para controlar el signo de los factores, en el simulador no se disponen y se deben considerar positivos, pues de lo contrario se pueden producir resultados poco coherentes, como ejemplo se ha comprobado que el multiplicador no debe ser negativo con este algoritmo, pues al restarle uno, no terminaría. Para multiplicar números negativos habría que analizar previamente el signo de cada factor y reprogramar el multiplicador cada vez. Se podría arreglar lo del multiplicador sumando uno en lugar de restar, pero obliga a reprogramar. Con el multiplicando no sucede ese problema pero hay que introducir manualmente el complemento a diez del numero y el resultado volverlo al sistema normal.

Una vez desarrollados los programas se hicieron las pruebas pertinentes y se analizaron los casos conflictivos que no se van a detallar, pero resultó curioso

ver que con algoritmos tan sencillos, algunos casos conflictivos eran simples y de sentido común. Como ejemplo baste mencionar la división, cuyo algoritmo es ir restando el divisor del dividendo hasta que solo queda el resto. Si se divide por cero, el proceso era obvio, se resta cero y el programa entraba en un bucle hasta que se parara a mano, sin más, no había desbordamientos ni se colgaba.

Otras cuestiones generales de los programas que es conveniente señalar. El orden y la limpieza pronto se hicieron necesarios para cablear los programas intentando evitar los cruces, así en este proceso de depuración se vio necesario tener reservado el primer acumulador con el uno, que lo enviaba negativo por la bandeja cuatro, la superior de datos y positivo por la bandeja de datos dos, para no cruzar cables se intentaba que los valores negativos circularan por encima y los positivos por debajo. Para seguir las explicaciones los acumuladores se numeran de izquierda a derecha, empezando por el uno.

Los programas se planteaban inicialmente con una etapa para cada operación, pero una vez que funcionaban se depuraban reagrupando etapas, dejando las mínimas en la fase dos y tres, también se reorganizaban las bandejas por las que se enviaban los datos o los puertos usados para quitar cruces o normalizar. En ambas fases, su primera etapa era *la bifurcación condicional*, que *siempre se lanzaba por ambos puertos A y S* al bus de programa, tomando solo el camino del que tuviera el uno en la línea uno, por ser negativo o menor que; eran concurrentes pero solo había una llave. Esto obligaba a reservar las bandejas de programa D y E para unirlas al bus de datos, conectando las etapas de programación solo sobre las bandejas A, B y C.

Seguidamente se van a detallar los programas, las explicaciones comunes o ideas básicas se irán realizando cuando se presenten, sin repetirse normalmente en los restantes programas. La programación del ENIAC como se ha venido señalando consistía en establecer las conexiones a los buses de datos y programa, colocando los interruptores en las posiciones adecuadas para que se ejecute el código previsto, por ello se han destacado con color diferente las tres fases, las conexiones de programa y las posiciones de los interruptores, pero otros aspectos como los buses de datos o los puentes no se han retocado, para evitar sobrecargar las imágenes ya bastante complicadas.

5.3. El multiplicador

La multiplicación se ejecuta con un algoritmo sencillo que suma el multiplicando al producto y resta uno al multiplicador, repitiendo estas operaciones hasta que el multiplicador valga cero. Como el ENIAC no discrimina el cero, realiza una iteración más para comprobar que el multiplicador es negativo y finaliza corrigiendo el sobrepaso con una restauración que resta el multiplicando al producto y suma uno al multiplicador.

El algoritmo necesita cuatro acumuladores, el común del uno, el multiplicando, el multiplicador y el cuarto acumulador para el resultado o producto.

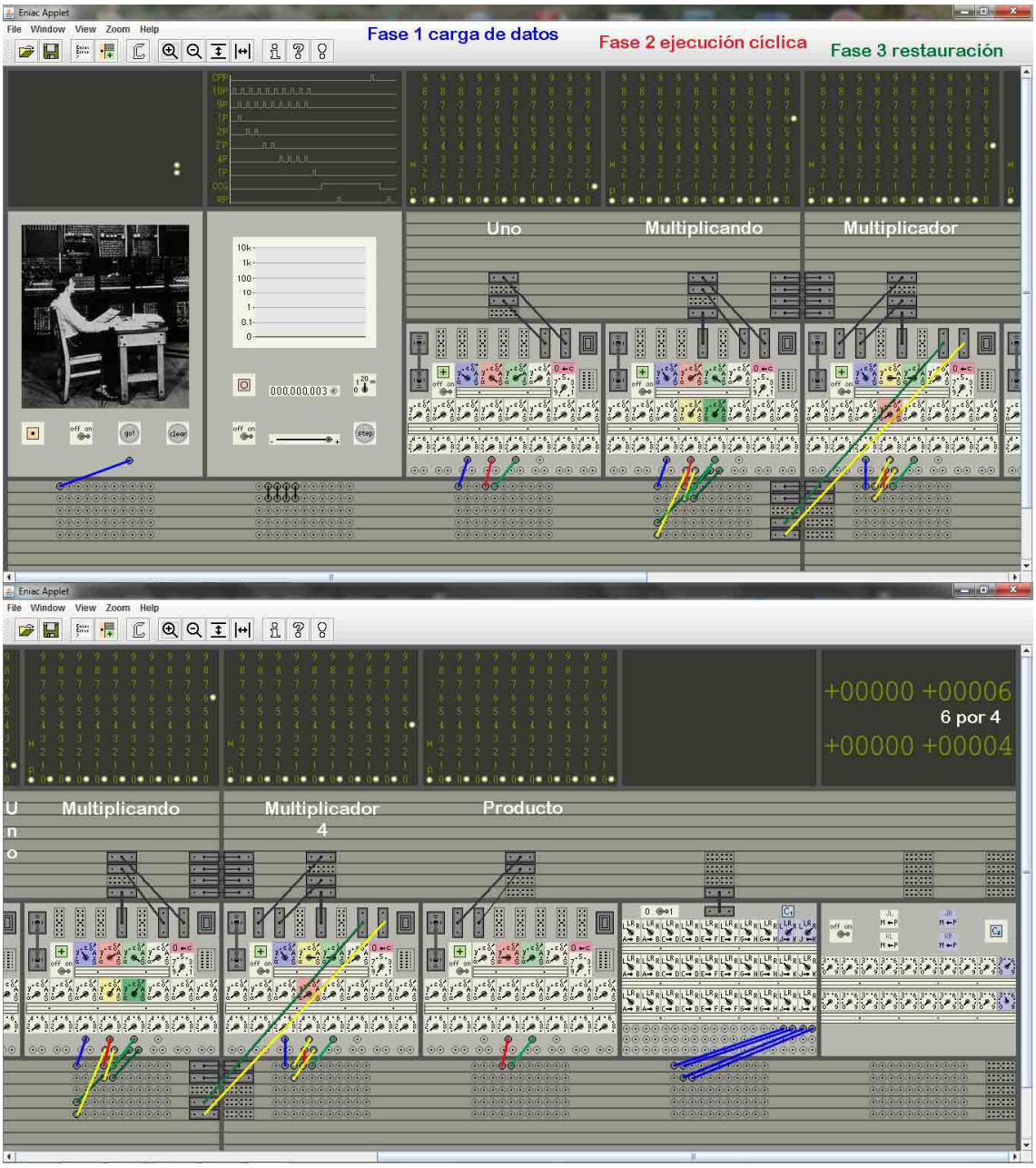


Imagen 56 – El multiplicador al inicio de un cálculo

El programa que implementa este algoritmo se muestra en la imagen 56, las fases están diferenciadas por color, en azul la *carga de datos*, en amarillo y rojo la *ejecución cíclica del algoritmo* y en verde oscuro y claro la *restauración*. La bifurcación condicional es la primera etapa de ambas fase dos y tres, va marcada en amarillo y en verde oscuro para diferenciarlas del resto. El procedimiento de cargar el uno en el primer acumulador consiste en fijar su interruptor de control de color azul como se ve en la imagen 56, con el interruptor de correcto borrado

en C y el otro interruptor de control en posición de recibir por un puerto que está desconectado. Los acumuladores usados deben ponerse a cero después de un cálculo si se quiere volver a repetir, por ello tienen puesto su interruptor de borrado en C, de esta manera pulsando manualmente el botón *clear* se borran.

El programa arranca cargando los datos y el multiplicador lanza la bifurcación condicional o *if,* espera un ciclo para determinar el camino a seguir y luego realiza todas las operaciones de la multiplicación paralelas en otro ciclo, repitiendo este bucle mientras se cumple la condición. Cuando el multiplicador llega a menos uno el programa acaba, corrigiendo antes la iteración de más con la restauración, suma uno al multiplicador y resta el multiplicando al producto dejando el producto exacto, estos detalles se pueden seguir en la imagen 57.

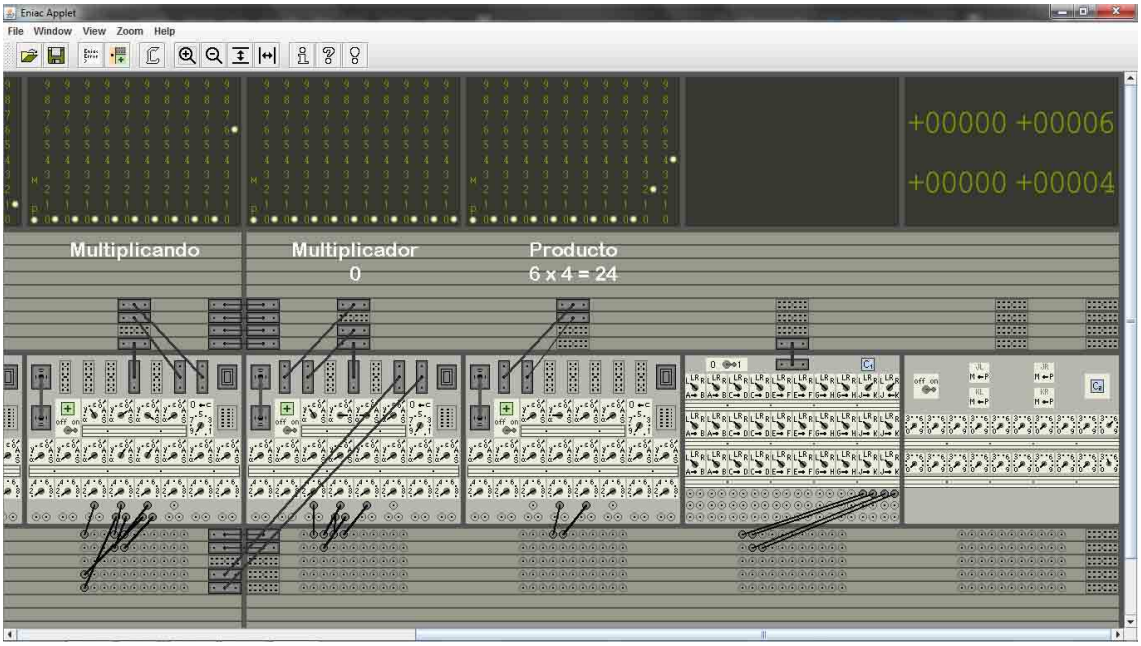


Imagen 57 – El multiplicador al finalizar

El coste o tiempo de ejecución es la talla del multiplicador por dos. Las operaciones de carga iniciales y la restauración se pueden ignorar pues no son significativas cuando la talla crece. Para reducir el coste, las operaciones se hacen paralelas. Reunir toda la fase en dos etapas y coordinar las fases, viene condicionado por las bandejas disponibles del bus de datos y los puertos de los acumuladores, sobre todo los de salida A y S. Se puede analizar el reparto de las cuatro bandejas disponibles, siguiendo la ruta de los datos en las tres fases. Las cuatro bandejas están distribuidas permitiendo usar compartidas y sin interferirse, una bandeja con el envío de datos desde el transmisor de constantes, tres en la fase dos y tres bandejas para la fase tres.

El código es similar al mostrado en la discriminación de magnitud.

```
/** Calcula el producto de M por N */
                              //Fase de carga de datos
  Acum1= 1, Acum2= M;             //Comienzo, pulso a línea 1
  Acum3= N;                       //Pulso a línea 2
                              //Fase de ejecución cíclica
  if(Acum3 >= 0){                 //Pulso a línea 3
                                  //Espera
  Acum4+= Acum2, Acum3-= Acum1;  //P. línea 4 y pulso línea 3
                              //Fase de restauración
  }                               //Espera
  Acum4-= Acum2, Acum3+= Acum1;  //Pulso a línea 5, final
```

Como aclaración, los programas presentan a la izquierda el código en un lenguaje tipo java y a la derecha en la columna comentada figuran las tres fases usadas y los pulsos que se corresponden a cada etapa en el simulador. El *if* en realidad lanza por el puerto A y S concurrentemente las dos opciones al bus de programa. No se ha concretado la instrucción espera, que podría ser un tipo de pausa, porque el ENIAC lo tenía establecido, era ejecuta cero durante un ciclo.

5.4. El divisor

La división es parecida al producto, su algoritmo consiste en restar el divisor al dividendo y se suma uno al cociente, repitiendo estas operaciones hasta que el dividendo cambie de signo a negativo, entonces se restaura ajustando el resultado por defecto, sumando el divisor al dividendo para obtener el resto y se resta uno al cociente. Este algoritmo necesita cuatro acumuladores, el del uno, el divisor, el dividendo o resto al finalizar y el cuarto acumulador del cociente.

El programa divisor arranca cargando dos datos, lanza la bifurcación condicional y espera un ciclo para determinar el camino a seguir, realizando en otro ciclo, como se puede ver en la imagen 58, todas las operaciones de la división en paralelo. Se repite el bucle mientras se cumple la condición. El programa finaliza en el momento que el dividendo pasa a negativo, realizando la restauración como se muestra en la imagen 59, que suma el divisor al dividendo para calcular el resto y resta uno al cociente, obteniendo el cociente por defecto.

Cuando la división es exacta, cosa poco frecuente, el algoritmo no necesitaría rebasar el dividendo para cumplir la condición. De todas formas el ENIAC realizaba una iteración de más, porque no discriminaba el cero del resto de positivos en el dividendo. El coste o tiempo de ejecución es la talla del cociente por dos, señalar que el cociente depende de dos números, siendo menor cuanto más próximos estén. Las operaciones de carga iniciales y la restauración se pueden ignorar pues no son significativas cuando crece la talla.

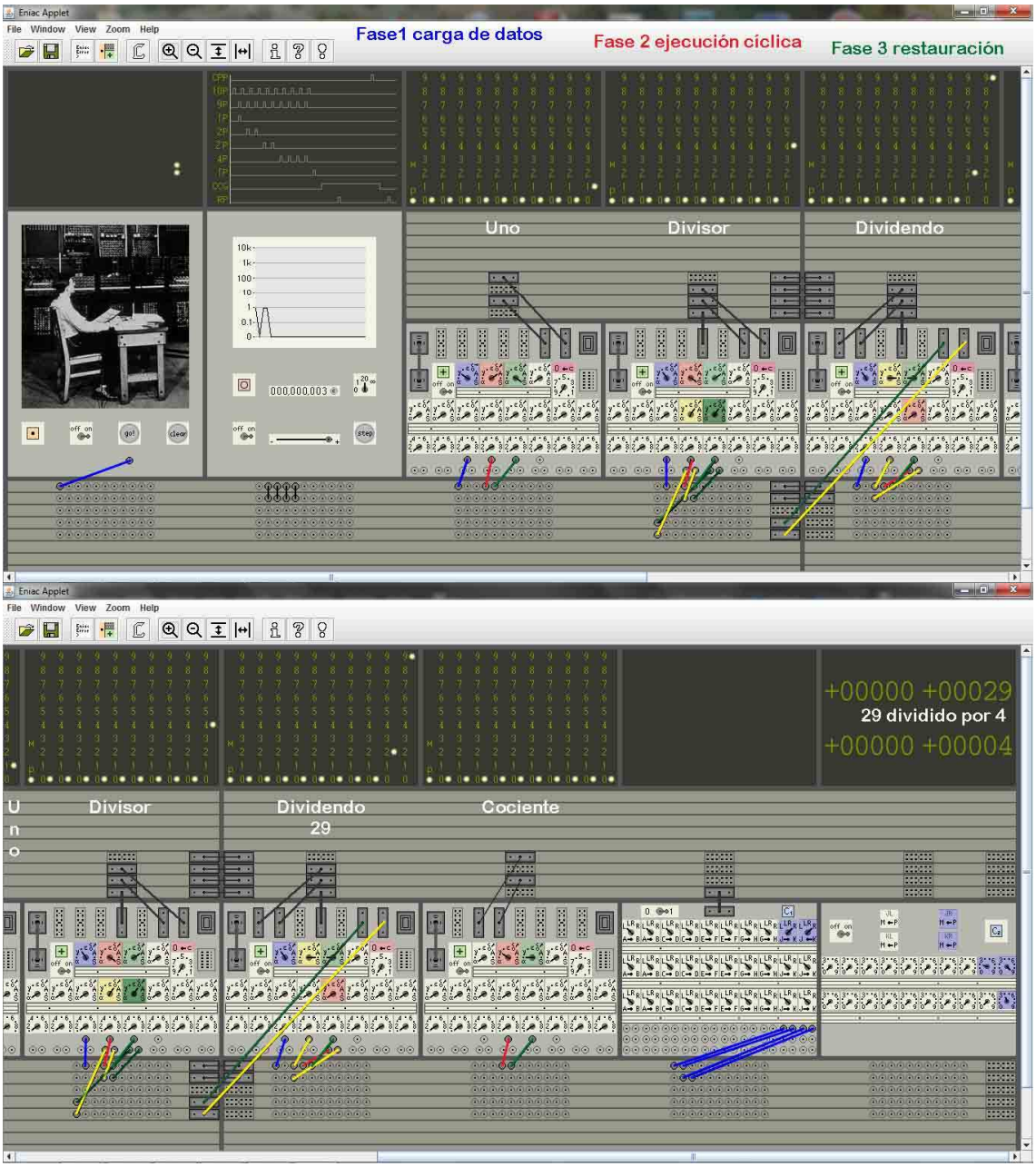


Imagen 58 – El divisor al inicio de un cálculo

El código del programa para dividir es:

```
/** Calcula el cociente y el resto de M dividido por N */
                              //Fase de carga de datos
  Acum1= 1, Acum2= N;             //Comienzo, pulso línea 1
  Acum3= M;                       //Pulso a línea 2
                              //Fase de ejecución cíclica
  if(Acum3 >= 0){                 //Pulso a línea 3
                                  //Espera
  Acum3-= Acum2, Acum4+= Acum1;   //P. línea 4 y pulso línea 3
                              //Fase de restauración
  }                               //Espera
  Acum3+= Acum2, Acum4-= Acum1;   //Pulso a línea 5, final
```

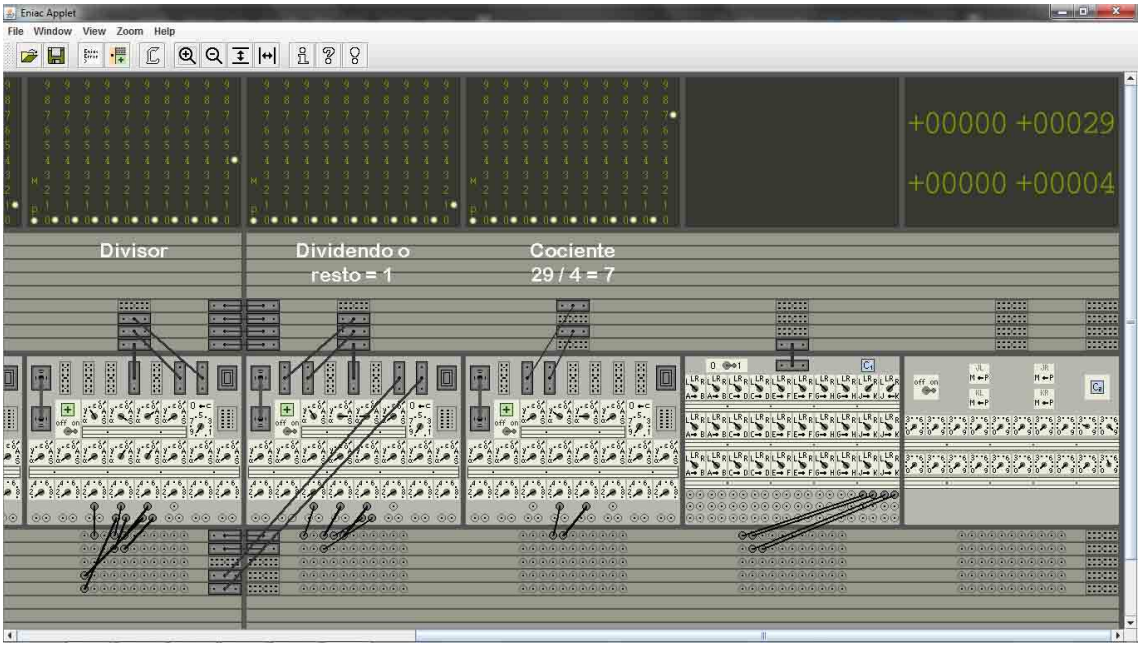


Imagen 59 – El divisor al finalizar

5.5. La raíz cuadrada

La raíz cuadrada tiene un esquema parecido a la división, salvando su diferencia con el "divisor" que no es constante, pues en la raíz cuadrada es una sucesión de números impares variable que empieza con el uno. Por ello se diseña el programa para calcular raíces sustituyendo el acumulador divisor por dos. Uno que contiene el dos para enviarlo formando la sucesión de impares y el otro acumulador, sobre el que se suma dicho dos para formar la serie creciente de impares, esta pareja de acumuladores cumplen una función similar al divisor. El resto de acumuladores se ajustan a las características de la raíz.

El algoritmo consiste sumar dos al impar menos uno al inicio, restar el impar resultante al radicando y se suma uno a la raíz cuadrada, repitiendo estas operaciones hasta que el radicando cambie de signo a negativo. Entonces se restaura ajustando el resultado por defecto. Se suma el impar al radicando, se resta uno a la raíz cuadrada y se restan dos al impar. Como ya se aprecia ha complicado un poco el algoritmo y además requiere en cada iteración otro ciclo para sumar dos al impar. Este algoritmo necesita cinco acumuladores, el del uno, el del dos, el impar, el radicando y el acumulador de la raíz cuadrada.

El programa raíz para el simulador se puede seguir en la imagen 60, arranca cargando el radicando y el uno en los acumuladores uno y dos, en otro ciclo se suma uno al acumulador dos y se resta uno al acumulador impar. De esta manera quedan preparados los tres acumuladores con uno, dos y menos uno. Se lanza un pulso al acumulador radicando que realiza la bifurcación condicional y

en el siguiente ciclo determina el camino a seguir, realizando ahora en dos ciclos todas las operaciones de la raíz cuadrada, pues hay que sumar dos al impar, restar el resultado al radicando y sumar uno a la raíz cuadrada. El bucle se repite mientras se cumple la condición, finaliza al pasar el radicando a negativo.

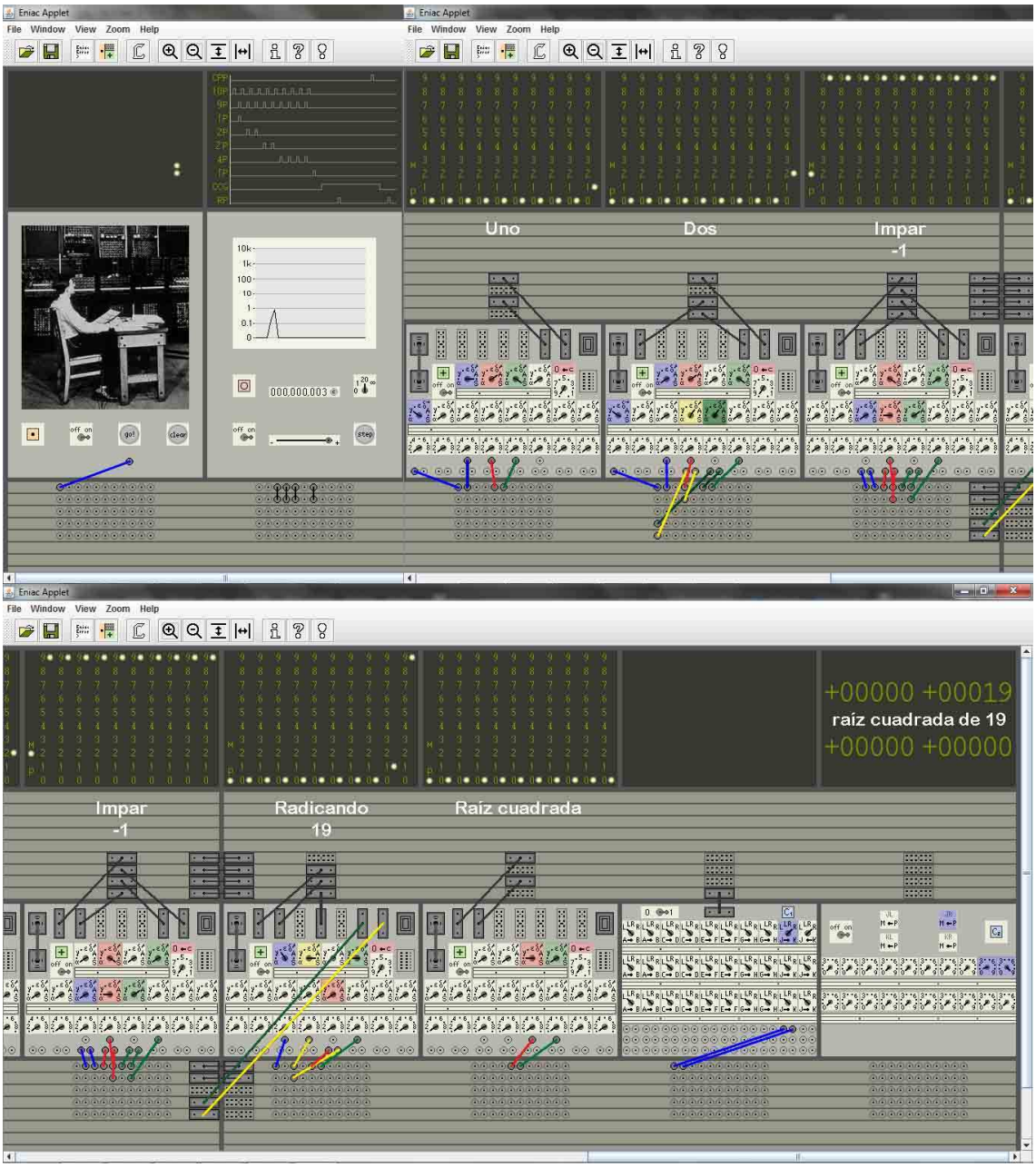


Imagen 60 – La raíz cuadrada al inicio de un cálculo

Entonces la bifurcación condicional ejecuta la fase de restauración. Esta también dura dos ciclos como se puede observar en la imagen 61, suma el impar al radicando dejando el resto de la raíz y suma uno a la raíz dejando en ese acumulador la raíz cuadrada por defecto, a continuación resta dos al impar, dejando en ese acumulador el último impar usado en la raíz cuadrada. El coste o

tiempo de ejecución es la raíz cuadrada de, la talla del número por tres, debido a la necesidad de modificar el impar cada vez. Las operaciones de carga iniciales y de la restauración se ignoraran pues no son significativas cuando la talla crece.

El código de la raíz cuadrada es:

```
/** Calcula la raíz cuadrada de M */
                              //Fase de carga de datos
  Acum1= 1, Acum2= 1, Acum4= M;  //Comienzo, pulso a línea 1
  Acum2+= Acum1, Acum3-= Acum1;  //Pulso a línea 2
                              //Fase de ejecución cíclica
  if(Acum4 >= 0){                 //Pulso a línea 3
                                  //Espera
  Acum3+= Acum2;                  //Pulso a línea 4
  Acum4-= Acum3, Acum5+= Acum1;  //P. línea 5 y pulso línea 3
                              //Fase de restauración
  }                               //Espera
  Acum5-= Acum1, Acum4+= Acum3;  //Pulso a línea 6
  Acum3-= Acum2;                  //Pulso a línea 7, final
```

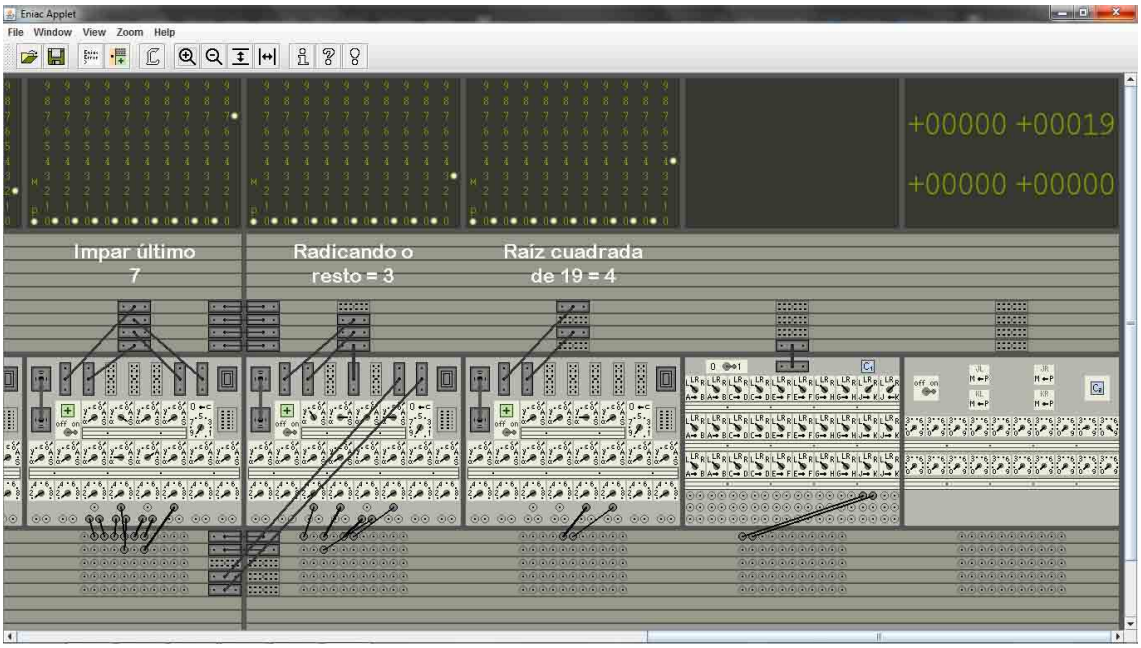


Imagen 61 – La raíz cuadrada al finalizar

Con estos programas el simulador recupera con limitaciones, las operaciones que podía realizar el ENIAC, necesitando cuatro o cinco acumuladores cada una. Ahora se ejecutan mediante sumas resultando más lentas. La división y la raíz cuadrada se calculan por defecto con restauración que obtiene el valor del resto, lo cual además de tener un carácter didáctico, permite aprovechar los datos, pues en la división el resto es la clase en

aritmética modular y en ambos casos se podrían aproximar calculando decimales a partir del resto, multiplicando por potencias de diez los factores, de manera similar a como se realizaba con los acumuladores desplazadores cuando se explicaron en el apartado 4.5.3 los algoritmos de estas operaciones.

5.6. El velocímetro

El velocímetro es un nombre corto y descriptivo para referirse a la prueba realizada al ENIAC en mayo de mil novecientos cuarenta y cuatro, donde conectaron dos acumuladores y midieron la rapidez a la que realizaba sumas, obteniendo la conocida velocidad media de cinco mil sumas por segundo. Este programa realiza una medición similar y permite estimar la velocidad a la que se ejecutan las sumas en cada equipo con el simulador.

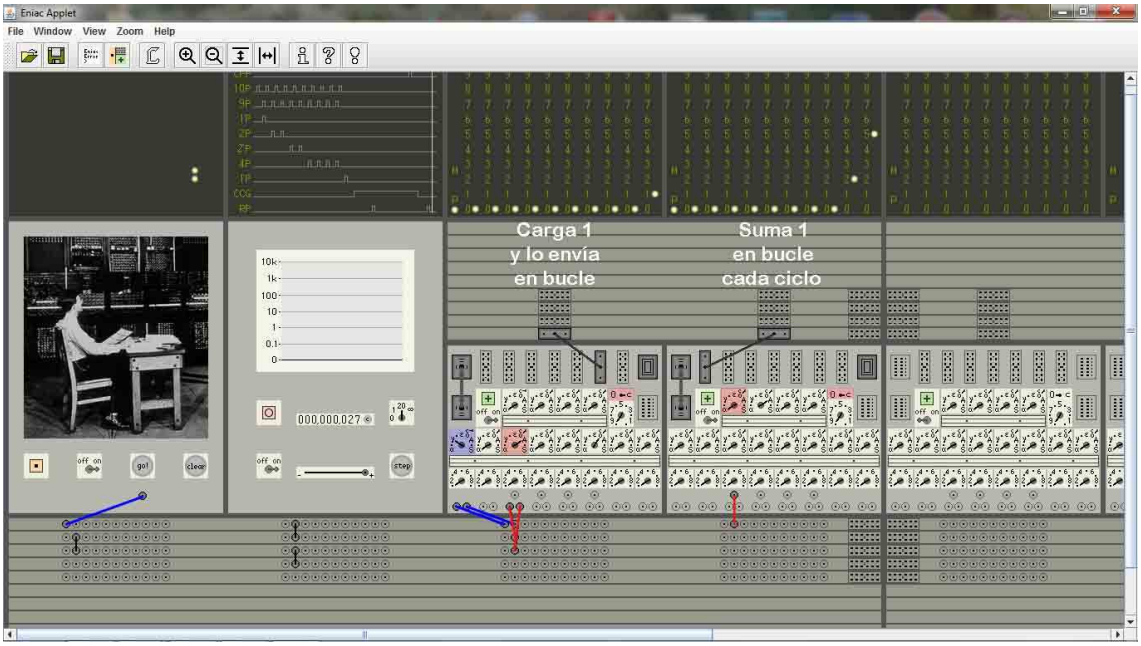


Imagen 62 – El velocímetro

En el velocímetro se han conectado dos acumuladores, el izquierdo en la fase de carga de datos recibe un uno y a continuación en la fase dos con una sola etapa, usando el bucle de cables que tiene conectado, lo envía al acumulador de la derecha que lo recibe y lo suma, repitiendo este bucle de manera continuada ambos, hasta que se detenga manualmente. Se suma uno para poder observar directamente en el acumulador de la derecha el número de sumas efectuadas, aunque ahora en la unidad de ciclos existe un contador que el ENIAC no tenía; se puede comprobar que ambas cifras coinciden con un desfase de dos pues el

contador emplea dos ciclos para cargar el uno y lanzar el pulso que arranca el bucle de envío y suma entre ambos acumuladores.

El día veintiuno de mayo pasado se probaron los programas del simulador en la sala de libre disposición de la etsinf, comprobando que funcionaban correctamente en otros equipos, tanto el simulador como los programas, y de paso se aprovechó para medir la velocidad de suma con dichos equipos. El proceso se realiza manualmente. Para reducir errores se han usado dos intervalos de medida con cuarenta y cinco y sesenta segundos, teniendo puesto el regulador de velocidad en la máxima posible, totalmente a la derecha.

Se realizó la medición lanzando el programa y parando a mano, justo en el segundo adecuado. Se tomaron tres datos considerando la mediana, diecinueve mil ciento veinte sumas y veintiocho mil doscientas dieciocho para cada intervalo, calculando una velocidad media de cuatrocientas veinticinco sumas por segundo para el de cuarenta y cinco segundos y cuatrocientas setenta con el de sesenta segundos, unas diez veces más lento que el ENIAC original.

Las velocidades medias difieren en cada intervalo, pero se achacó entre otras a que el simulador en realidad es un programa de java que también introduce sus variaciones, así por ejemplo hubo que tomar cuatro mediciones pues la primera normalmente era inferior a las otras tres y se descartaba. También se ha observado que el simulador parece hacer las sumas con mayor velocidad en programas simples, yendo más lento en programas complejos.

5.7. La suma de una serie de números

El cálculo de la suma de una serie de números hasta el uno, parte de un esquema parecido al producto, la diferencia radica en que no se suma el mismo multiplicando al resultado, sino que va sumando por decirlo de forma simple, el multiplicador decrementado en uno. Otro detalle importante es que resulta más sencillo sumar la serie desde el número a cero que a la inversa. Para sumar la serie de números, se usan cuatro acumuladores, como se ve en la imagen 63, en el primero se pone un uno y se carga el número inicial en los restantes tres acumuladores. A continuación comienza la fase de iteraciones, se resta uno en los acumuladores dos y tres y se suma el resultado en el cuarto acumulador.

Se repite el bucle de ejecuciones cíclicas hasta que se cumple la condición cuando el dato del acumulador tres alcanza el cero, con este simulador cuando alcance menos uno con una iteración de más, como se puede ver en la imagen 64. Llamaba la atención tener que ir restando uno al mismo dato en los dos acumuladores porque solo tienen un puerto de salida A y se necesita enviar la resta a dos sitios, por el *bus de datos* desde el acumulador dos al resultado y

además desde acumulador tres se envía por otro camino al *bus de programación* para que la bifurcación condicional decida si continúa o finaliza.

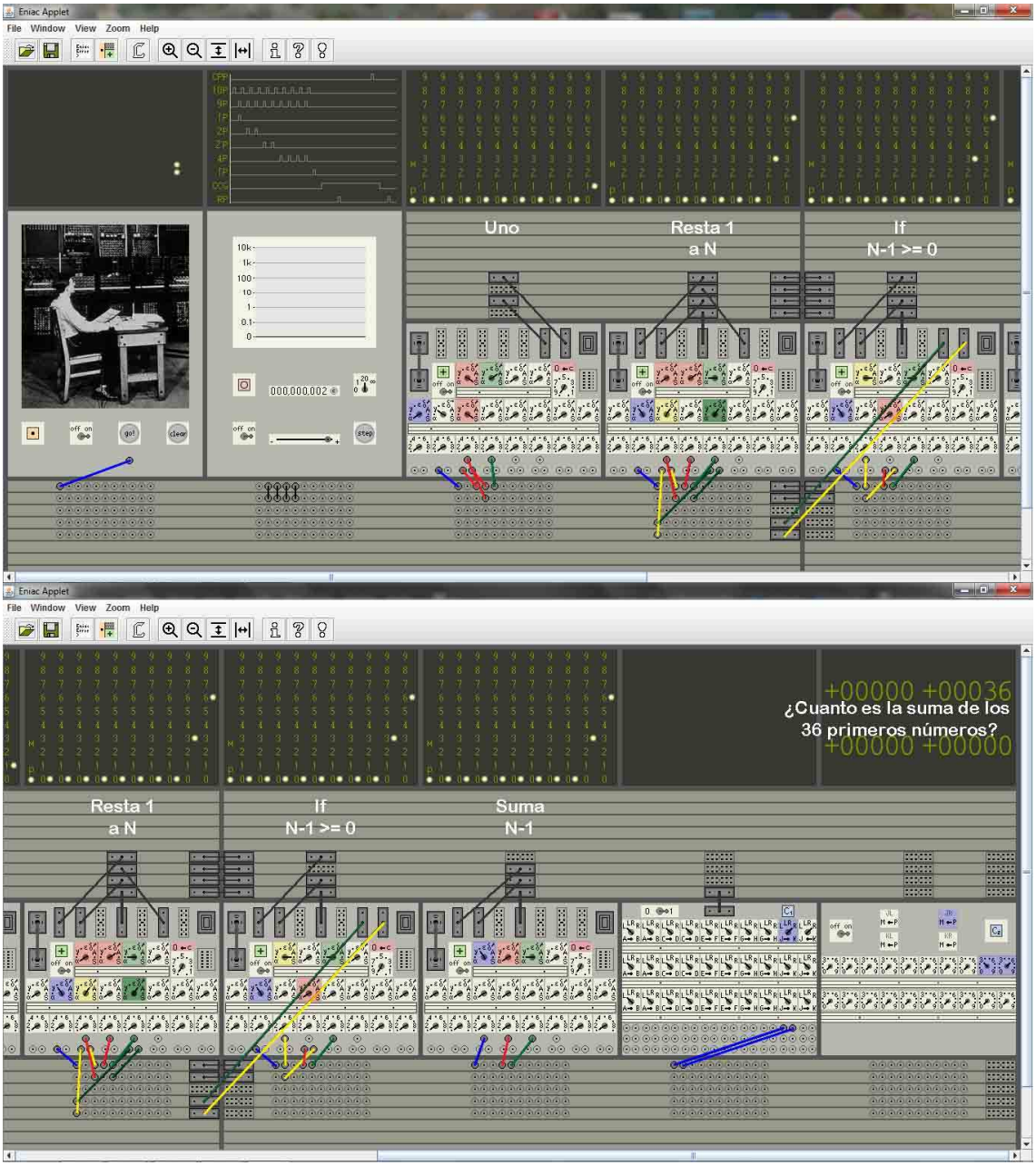


Imagen 63 – Suma de los 36 primeros números, inicio

El código de este programa con cuatro acumuladores era:

```
/** Calcula la suma de los N primeros números */
                              //Fase de carga de datos
  Acum1= 1, Acum2= N,              //Comienzo, pulso a línea 1
  Acum3= N, Acum4= N;
                              //Fase de ejecución cíclica
  if(Acum3 >= 0){                  //Pulso a línea 2
```

```
                                    //Espera
Acum2-= Acum1;                      //Pulso a línea 3
Acum4+= Acum2, Acum3-= Acum1;  //P. línea 4 y pulso línea 2
                              //Fase de restauración
}                                   //Espera
Acum2+= Acum1,  Acum3+= Acum1, //Pulso a línea 5, final
Acum4+= Acum1;
```

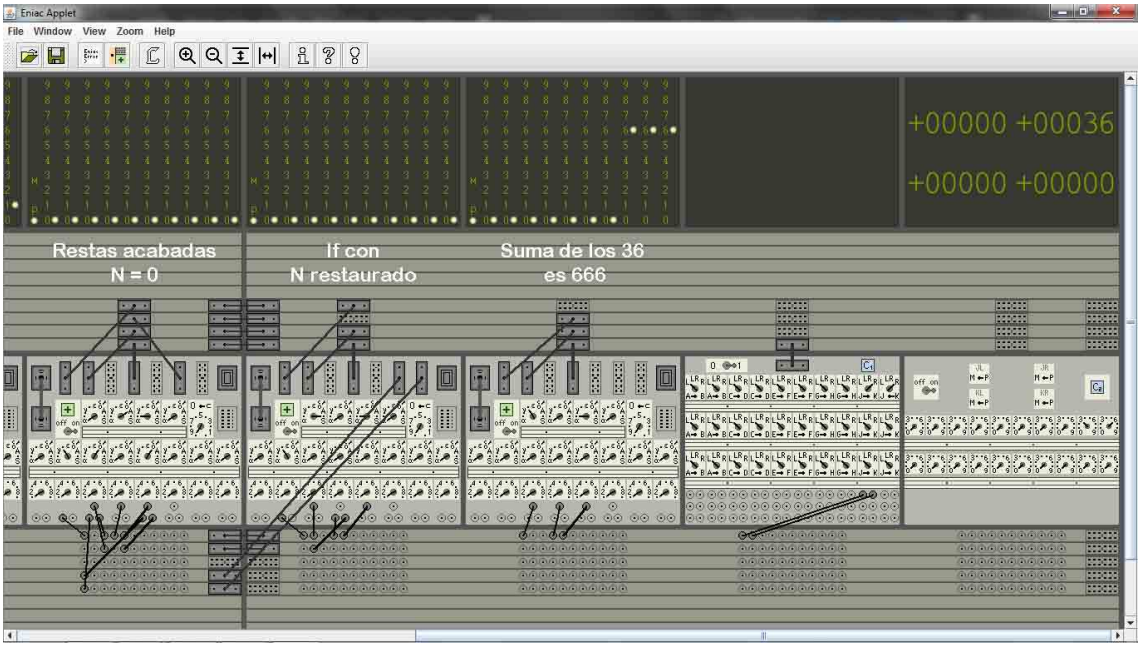


Imagen 64 – Suma de los 36 primeros números, finalizado

Luego, este programa se depuró para realizar el proceso con tres acumuladores, la diferencia residía en poder usar el dato del acumulador dos con el número menos uno, por el bus de datos para sumarlo al resultado y a la vez por el bus de programa comprobar la bifurcación, mediante una derivación como se muestra en la imagen 65. Siguiendo el recorrido se observa como el acumulador dos envía su dato a ambos sitios dando amplio un rodeo.

Esta operación tan sencilla, costó trabajo, pues no se lograba hacer funcionar el programa conectando el tercer acumulador por la bandeja de programa D, donde llegaba el dato desde A con el resultado de la resta, hasta que se conectaron al revés, primero al acumulador tres, luego se continuaba derivado y se conectaba a la bandeja de programa D. Con esto resuelto se rehízo el código con tres acumuladores, introduciendo algunas adaptaciones.

El código modificado para tres acumuladores es como sigue:

```
/** Calcula la suma de los N primeros números */
                              //Fase de carga de datos
  Acum1= 1, Acum2= N;               //Comienzo, pulso a línea 1
```

```
                                    //Fase de ejecución cíclica
if(Acum2 >= 0){                         //Pulso a línea 2
(Acum3+= Acum2;)                        //Espera
Acum2-= Acum1;                          //P. línea 3 y pulso línea 2
                                    //Fase de restauración
}                                       //Espera
Acum2+= Acum1, Acum3+= Acum1;   //Pulso a línea 4, final
```

El coste o tiempo de ejecución es la talla del número por dos. Aunque en la opción con cuatro acumuladores, debido a la repetición de la resta se ejecutaba con una etapa más por iteración, en la opción con tres acumuladores ya se corrige dejándolo en la talla por dos. Las operaciones de carga iniciales y la restauración se pueden ignorar pues no son significativas cuando la talla crece.

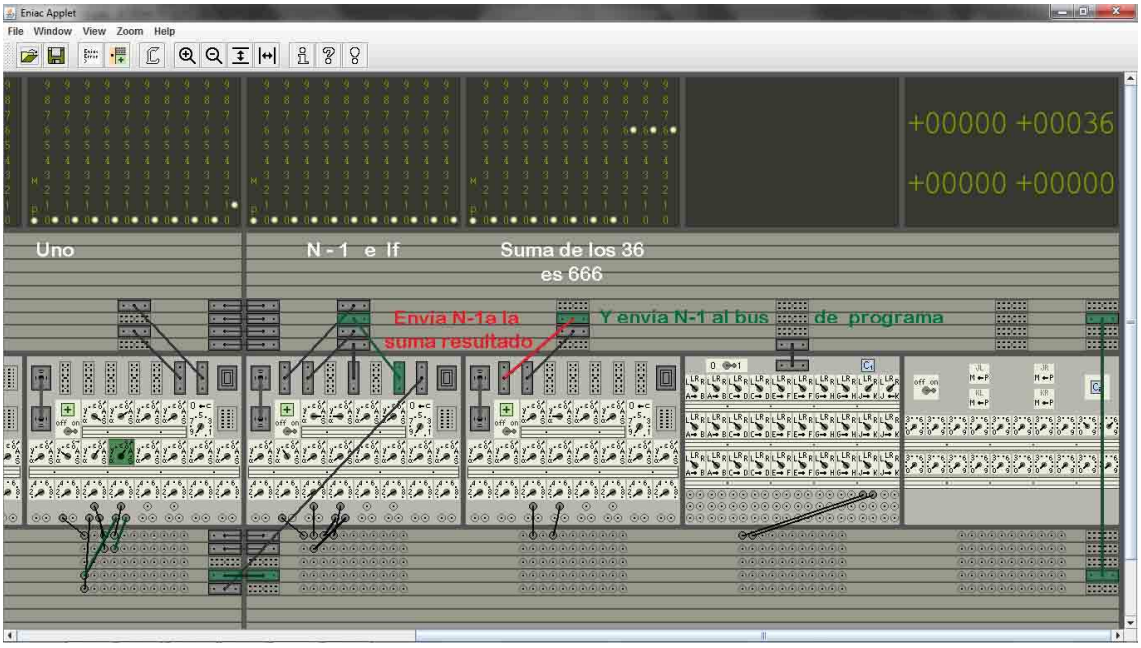


Imagen 65 – Suma de ene números con tres acumuladores, detalle

5.8. Elevar un número al cubo

El fundamento matemático es la suma de los términos de una progresión aritmética, cuyo valor es el producto de ene por la suma del primer y el último término de la progresión, dividido por dos. Ahora se ha estudiado el proceso inverso, desde el resultado se deduce cuánto debe ser el primer término de la serie y la razón, para que la suma de los ene términos sea ene al cubo. La solución obtenida fue que el primer término tiene que ser el mismo número y la razón el doble del número. Se puede comprobar calculando dicha suma. En esta progresión de razón 2N, el último término es N + 2N*(N-1) que resolviendo da

N+ $2N^2$ - 2N y simplificando es $2N^2 – N$, si le sumamos el primer termino N , la suma vale $2N^2$. Con lo cual la suma de la progresión vale $N*2N^2/2$, que es N^3.

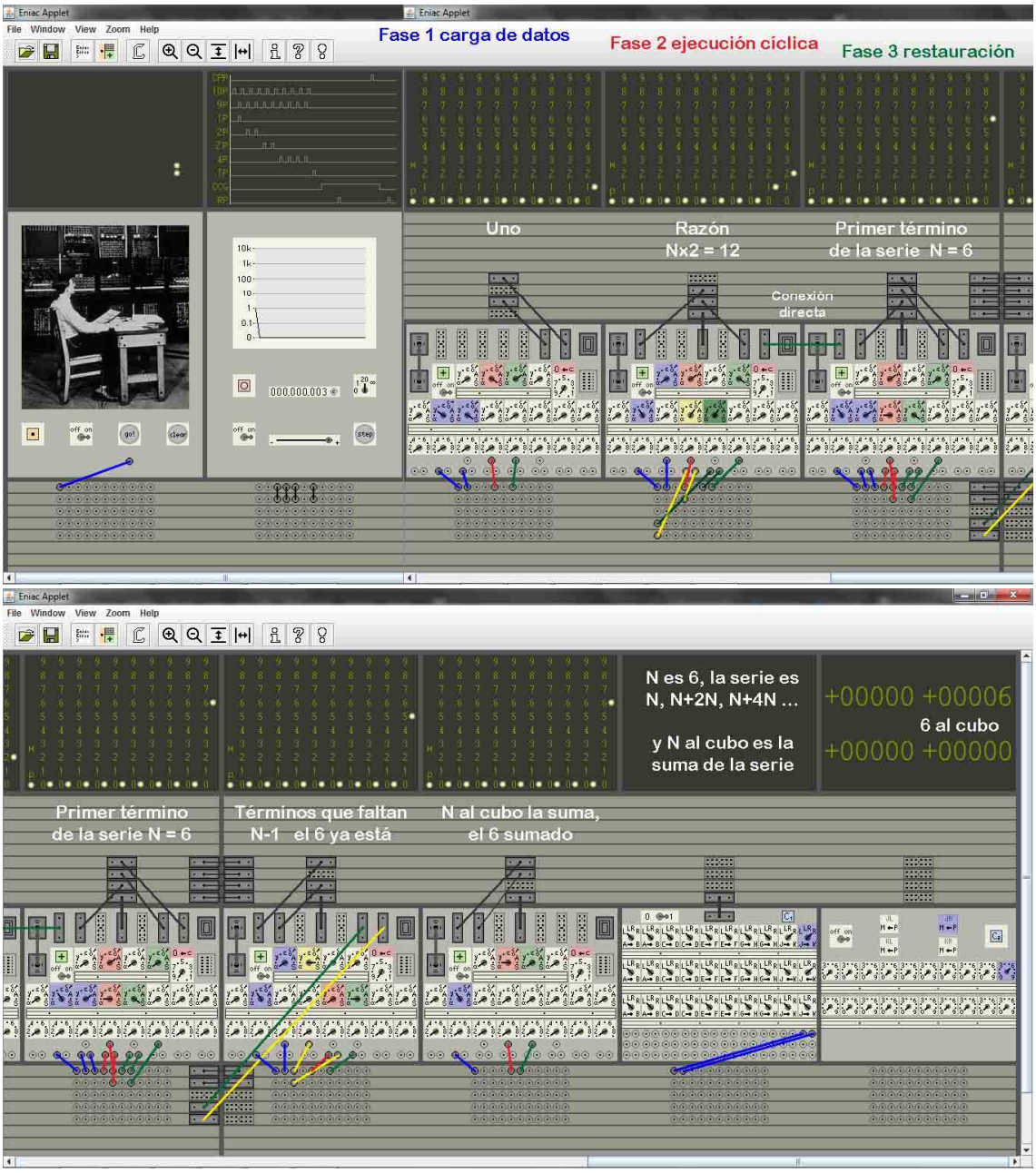


Imagen 66 – Elevar un número al cubo, inicio

El programa que calcula el cubo de un número, parte del esquema de la raíz cuadrada con unas adaptaciones. Las principales diferencias estriban en que esta serie ya no son los impares sucesivos y en el resultado, que tampoco es las veces que se itera, sino la suma de los elementos de la serie. Por ello hay que programar el desarrollo de la serie e ir sumando al resultado los ene términos. Con estos planteamientos se desarrolla el programa, como se puede observar en la imagen 66, con cinco acumuladores, para el uno, la razón, para el valor del

término de la serie, el cuarto, los términos que faltan de la serie, son las veces a iterar y el quinto acumulador con la suma de la serie, es el número al cubo.

El programa del simulador comienza con la carga de datos, en la primera etapa se carga el uno al primer acumulador y el número a elevar al cubo en los otros cuatro acumuladores. Se indica que como es necesario usar otra etapa para obtener la razón de la serie en el segundo acumulador, se aprovecha para efectuar ya la primera iteración, cargando el primer término en el quinto acumulador del resultado, que ya se efectuó en la primera etapa y se resta uno al cuarto acumulador con las veces a iterar, quedando con ene menos uno. De esta manera quedan los acumuladores listos, como aparecen en la imagen 66.

A continuación se lanza un pulso a la línea tres para arrancar la segunda fase de ejecución cíclica, con la bifurcación condicional comprobando que la condición se cumple, términos que faltan es mayor que cero. Se va repitiendo el bucle de sumar la razón al término de la serie, añadir el resultado a la suma de la serie y restar uno a términos que faltan. Así hasta que este llega a menos uno, donde la bifurcación condicional toma el camino de finalizar con restauración por haber ejecutado una iteración de más, pues se debería finalizar al llegar términos que faltan a cero. Los acumuladores quedan como se ve en la imagen 67, con el último término de la serie usado y el resultado del cubo.

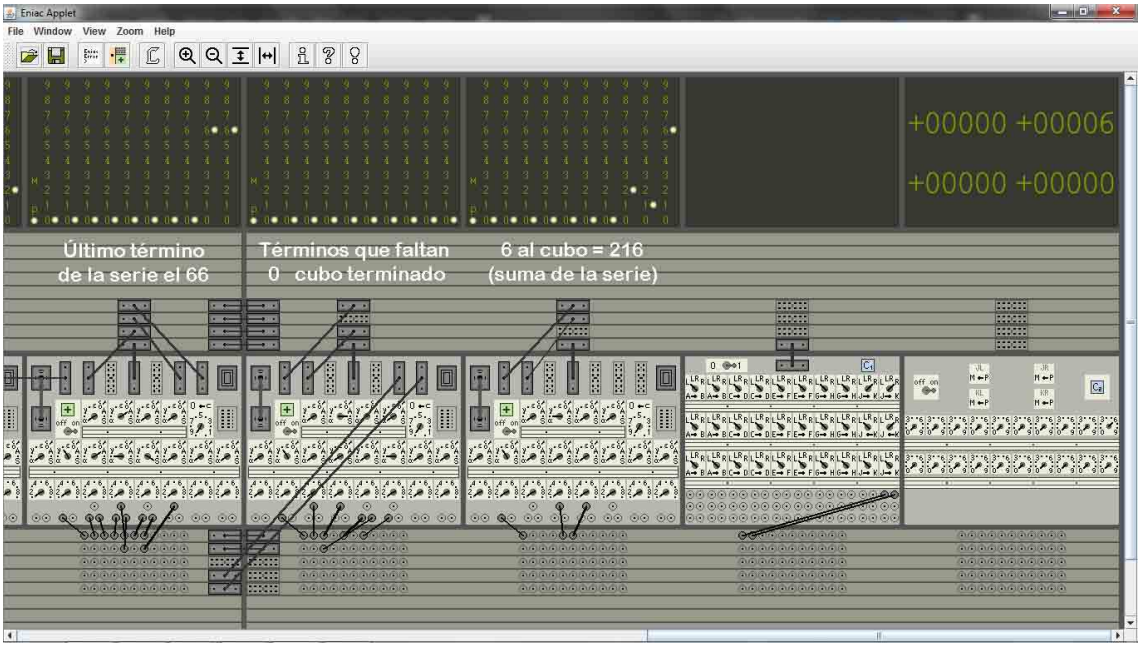


Imagen 67 – Elevar un número al cubo, resultado

El código de este programa es:

```
/** Calcula el cubo de N */
                              //Fase de carga de datos
  Acum1= 1, Acum2= N,             //Comienzo, pulso a línea 1
  Acum3= N, Acum4= N, Acum5= N;
  Acum2+= Acum3, Acum4-= Acum1;  //Pulso a línea 2
                              //Fase de ejecución cíclica
  if(Acum4 >= 0){                 //Pulso a línea 3
                                  //Espera
  Acum3+= Acum2;                  //Pulso a línea 4
  Acum5+= Acum3, Acum4-= Acum1;  //P. línea 5 y pulso línea 3
                              //Fase de restauración
  }                               //Espera
  Acum5 -= Acum3;                 //Pulso a línea 6
  Acum3 -= Acum2, Acum4+= Acum1; //Pulso a línea 7, final
```

El coste o tiempo de ejecución es la talla del número por tres, debido a la necesidad de modificar el término de la serie cada vez. Las operaciones de carga iniciales y de la restauración se pueden ignorar, no son significativas al crecer la talla. Para finalizar en este programa, como se observa en la imagen 66 superior, bajo el texto *conexión directa,* se pone de manifiesto en la etapa siete la escasez de puertos y buses de datos cuando el acumulador dos tiene que enviar restando la razón al acumulador 3, para restaurar el último término, concretamente en *Acum3 -= Acum2*. El problema era que el acumulador dos solo podía conectarse a la cuarta bandeja de datos superior, pero en esa bandeja el acumulador tres envía y no puede recibir. Después de darle vueltas sin resultado, se optó por conectarlos directamente *en el simulador,* opción obvia que no se contempla en los textos, aunque habría que saber si el ENIAC original permitía conectarse así.

5.9. Cálculo de una trayectoria

Con estas aplicaciones resueltas se abordó el cálculo semiautomático de una trayectoria en el espacio sin rozamiento. A partir de las fórmulas físicas se redujeron los cálculos a seis multiplicaciones y una división que en principio tampoco era factible pues hacían falta veintiocho acumuladores, dado que cada multiplicación o división usa cuatro acumuladores, aunque después de una serie de medidas, compartiendo acumuladores y rediseñando los multiplicadores se consiguió un programa que calcula la trayectoria.

El funcionamiento combinado es el aspecto que se va a tratar aprovechando este programa, pues el ENIAC era capaz de realizar cálculos de manera paralela o sucesiva, con sus acumuladores que podían almacenar resultados intermedios y comunicarse para transferir automáticamente datos o

resultados entre ellos. Ahora se va a insistir este concepto para buscar y analizar las interacciones entre los grupos de acumuladores, además del propio cálculo.

En el punto 5.2 sobre la programación se definieron las *fases y etapas* de los programas para ayudarnos a estructurar los algoritmos. Ahora se añade otro concepto para comprender mejor las interacciones entre grupos de acumuladores. Se define de manera particular la *máquina* como el conjunto de acumuladores que realizan un programa, por ejemplo la máquina de dividir serían los cuatro acumuladores que calculan una división, la máquina se comporta como una caja negra que recibe las entradas, el dividendo y el divisor y calcula las salidas el cociente y el resto, evitando tratar detalles internos.

Así cuando en el simulador se implementen juntas dos máquinas de multiplicar y dividir, por ejemplo para realizar otro programa general que haga un cálculo pasando datos de una a otra, usaremos este concepto para explicar más fácilmente la interacción entre máquinas sin necesidad de bajar al funcionamiento interno, ya explicado en los programas particulares, es como una capa superior para describir comunicaciones entre grupos. Las tres rutinas enlace en el código del programa trayectoria contendrán esta información.

Para programar el ENIAC se dijo en el punto 4.4 al tratar los buses, que se partía de un problema de cálculo a realizar, el cual se materializa en una serie de ecuaciones matemáticas. Estas ecuaciones se debían reorganizar en una secuencia de operaciones básicas que el ENIAC podía ejecutar. Sobre esta secuencia se planificaba el recorrido y almacenamiento de los datos iniciales, resultados intermedios y finales. Esto es lo que hace el cálculo de la trayectoria.

El problema, calcular la trayectoria del movimiento de proyectiles en el espacio sin rozamiento. Es un problema físico que a partir de la velocidad del proyectil *v*, el ángulo de tiro *a* con su *seno* y *coseno* y el valor de la aceleración de la gravedad *g*, calcula con unas fórmulas el tiempo de vuelo del proyectil *tv*, desde que sale de la boca de fuego hasta que toca el suelo, la flecha o altura máxima *h,* y el alcance *xmax,* o distancia máxima recorrida en su vuelo.

El proyectil sale disparado con su velocidad inicial de subida pero la gravedad terrestre lo frena con su aceleración hacia el suelo, la altura máxima *h*, se logra cuando velocidad de subida es igual a la originada por la gravedad, esto determina el tiempo de subida *ts*, y además si consideramos que no hay rozamiento, el tiempo de bajada *tb* será el mismo. Con ello tenemos el tiempo de vuelo como la suma de ambos *tv = ts+tb*. Por tanto la altura máxima es la alcanzada en la subida, medio tiempo de vuelo y la distancia máxima *xmax*, es la recorrida durante todo el tiempo de vuelo, tanto de subida como de bajada.

Con los datos iniciales *v, sen(a), cos(a) y g*, partimos de las fórmulas físicas mostradas debajo que hay que reconvertir en una secuencia de operaciones

básicas del ENIAC que optimicen los medios, tratando de aprovechar los cálculos comunes y operar de la forma mejor.

$$\mathbf{ts} = \frac{v*sen(a)}{g}, \qquad \mathbf{h} = \frac{(v*sen(a))*(v*sen(a))}{2*g}, \qquad \mathbf{xmax} = (v*cos(a))*tv$$

Antes de calcular se analizan los elementos comunes. Dos productos se repiten bastante (v*sen(a)) y (v*cos(a)); además tienen factor común v. Se necesitan ocho acumuladores para realizarlos, pero si se agrupan en un solo multiplicador con dos multiplicandos, solo harían falta seis acumuladores. La idea consiste en coger el producto (sen(a)*v) y acoplarle los dos acumuladores del coseno, su multiplicando y su resultado, así se podría calcular (cos(a)*v) aprovechando las iteraciones del producto seno por v, pues ambos tienen el mismo multiplicador, esto será la máquina denominada multiplicador doble.

Por otro lado hay que dividir (v*sen(a)) por g y por 2*g, resulta mejor hacer una sola división si se divide v*sen(a) por 2*g y luego ts se obtiene sumando dos veces ese resultado, pues así solo necesita otro acumulador. Para multiplicar hasta por nueve, se hace directamente con otro acumulador fijando sus interruptores de control y enviando esas veces el dato del primero al otro.

Recopilando las formulas anteriores con los datos iniciales, hay que calcular tres valores, el tiempo total de vuelo, la altura máxima y el alcance horizontal.

$$\mathbf{tv} = ts+tb = 2*ts = 2*\left[\frac{sen(a)*v*2}{2*g}\right] = \mathbf{4}*\mathbf{\left[\frac{sen(a)*v}{2*g}\right]}$$

$$\mathbf{h} = \textit{(sen(a)*v)} * \mathbf{\left[\frac{sen(a)*v}{2*g}\right]}$$

$$\mathbf{xmax} = (cos(a)*v)*tv = \mathbf{4}*\textit{(cos(a)*v)}*\mathbf{\left[\frac{sen(a)*v}{2*g}\right]}$$

Antes se habían realizado dos operaciones.
sen(a)*v y cos(a)*v, en un multiplicador doble con seis acumuladores.
sen(a)*v/(2*g), que es el ts/2 o tv/4, en un divisor con cuatro acumuladores.
Falta por calcular con otro multiplicador doble *(cos(a)*v)* * (sen(a)*v/2g) y *(sen(a)*v)* * (sen(a)*v/2g), que necesita, seis acumuladores.
El tiempo de vuelo tv requiere un acumulador para multiplicar por cuatro.
Y xmax requiere otro acumulador para multiplicar por cuatro.

En total se necesitan dieciocho acumuladores, como siguen faltando dos, se puede compartir el primer acumulador que contiene el uno, por los dos productos dobles y el cociente, es igual en todos y se logra activar para las tres operaciones. De este modo se ahorran dos y con los dieciséis acumuladores se pueden realizar los seis productos y la división, montando tres máquinas.

El programa trayectoria monta tres máquinas y dos acumuladores que multiplican por cuatro. El flujo comienza en el transmisor de constantes que almacena los datos iniciales, *seno de a*, *coseno de a*, ambos con tres decimales, la *velocidad* inicial del proyectil que puede fijarse de cero a novecientos metros por segundo para no rebasar los diez dígitos de los acumuladores, y el valor *ciento noventa y seis* que corresponde a la constante "2*g" con un decimal. El ENIAC no usaba mecanismos de control y el interruptor de cifras significativas explicado en el apartado 4.5.1, no tenía coma para separar las cifras decimales e introducía un cinco que producía resultados digamos peculiares.

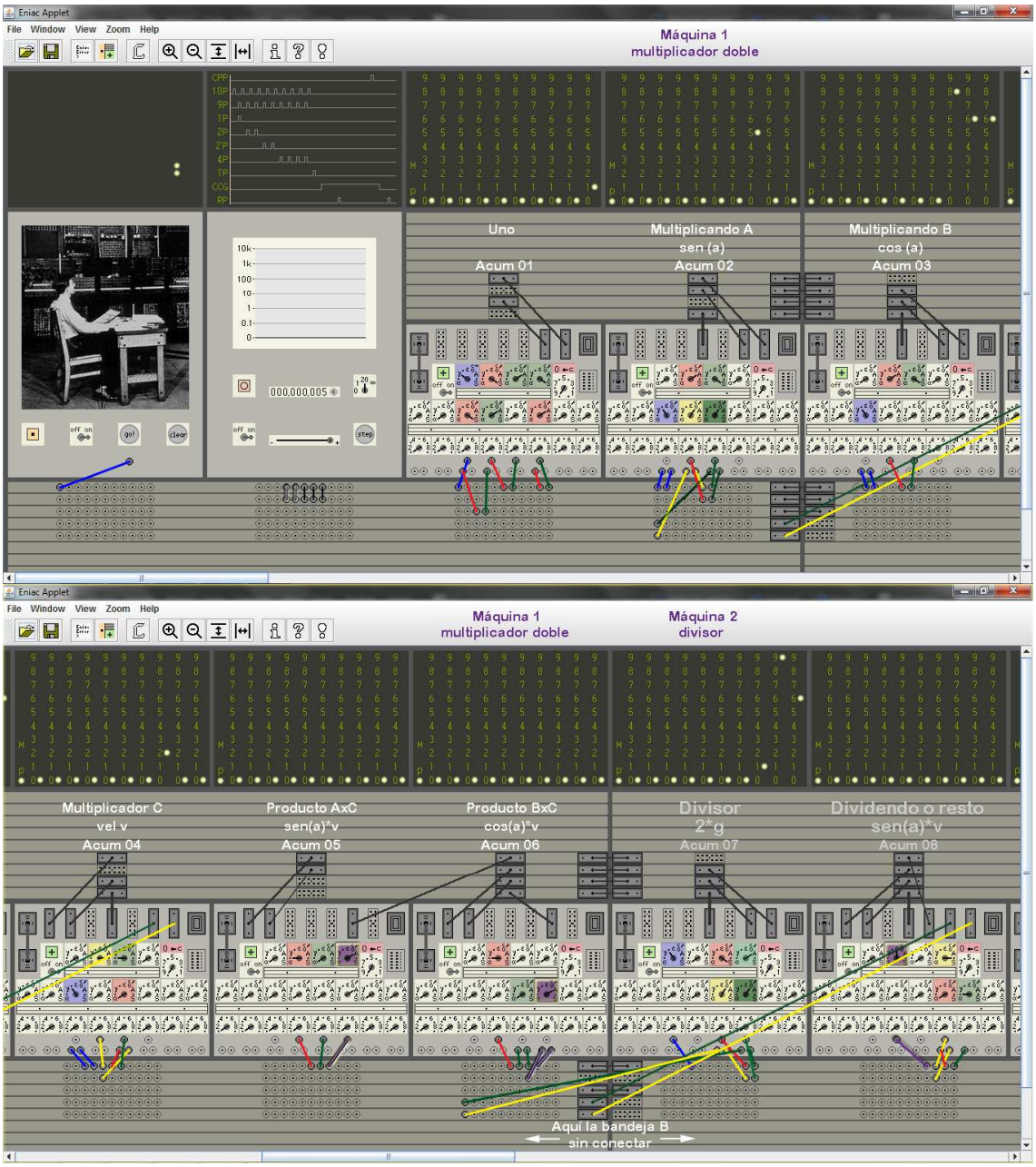


Imagen 68 – Trayectoria con el ENIAC primera parte, inicio

Por otro lado era importante tener controlado el numero de decimales de entrada y salida en cada operación, pues el ENIAC sin coma decimal, multiplicaba todo y había que saber quiénes eran decimales. Se confeccionó una hoja de cálculo que daba los resultados y servía de comprobación. Así se determinó la distribución adecuada, detallada gráficamente en la imagen 70 y 71, con el número de dígitos significativos, decimales incluidos, en cada factor para obtener los resultados con precisión, en tiempo oportuno y sin rebasar los diez dígitos, realizando luego tandas de comprobación en el simulador. De esta forma los resultados finales consiguen el tiempo con dos decimales, así como la altura y el alcance con cinco decimales en sus respectivas cifras significativas.

Imagen 69 – Trayectoria con el ENIAC segunda parte, inicio

Hubiera sido deseable un decimal más en el seno, coseno y el tiempo de vuelo, pero se descartó pues al terminar solo quedaban tres dígitos para la parte entera de la altura y el alcance, que con siete decimales estaban muy desproporcionados y además multiplicaba el tiempo de ejecución por unos diez. Respecto a los tiempos de duración del programa, para un cálculo medio considerando velocidades de quinientos metros por segundo, se realizan más de tres mil sumas en las máquinas, con un coste de dos ciclos por suma.

Una vez vistos los datos iniciales y las cifras decimales a usar, se describe de manera somera el multiplicador doble, cuya idea de funcionamiento ya se ha explicado pero no se había concretado. En la imagen 68 los acumuladores uno a seis ambos inclusive forman el citado multiplicador doble, cuya estructura es igual al multiplicador simple, con la diferencia de que este caso tiene dos multiplicandos y dos resultados aprovechando el multiplicador común. El coste o tiempo de ejecución es el mismo, la talla del multiplicador por dos, dado que se pueden resolver los dos productos en paralelo.

La estructura del programa trayectoria, se puede seguir completa en las imágenes 68 y 69, con la carga de datos inicial y en las imágenes 70 y 71 con los resultados obtenidos y el estado de los acumuladores. En dichas ilustraciones se sigue la norma de colores y organización de los programas vistos hasta ahora, con sus tres fases y las etapas necesarias, pero se han añadido las relaciones entre máquinas en las rutinas enlace, pasándose datos y la activación entre ellas, que se han destacado en relieve de color morado y amarillo con los interruptores de control y las conexiones al bus de programa de los acumuladores implicados.

En este programa realizado con la versión *Extended large* del simulador se utilizan todos los medios disponibles y aparecen problemas de escasez de recursos que se solucionan sobre la marcha organizando conexiones o aislando partes. El bus de programa se distribuye entre la programación de etapas con tres bandejas A, B y C y para las tres bifurcaciones condicionales se reservan las dos bandejas D y E, aisladas y sin interconectar los grupos de cuatro elementos.

Las tres primeras bandejas son para programar al conectarles los pulsos de activación, teniendo la bandeja B puenteada pues se requieren dos conexiones constantemente, por ello se puede decir que los pulsos comienzan en la bandeja A línea uno, o A-línea 1, a once y siguen por la bandeja C línea uno a once, o C-línea 11. Lo cual permite realizar hasta veintidós etapas. Las líneas cinco y seis se solapan en la bandeja A y C, por ello se ha cortado la interconexión de la bandeja B a la altura de los acumuladores seis y siete, aislando una parte de la otra y permitiendo hacer los puentes sin que se interfieran. Los puentes están en la unidad de ciclos y en el panel dos del transmisor de constantes.

El programa se ejecuta en las máquinas de manera secuencial, de izquierda a derecha, pues van necesitando los resultados anteriores, aunque dentro de

cada máquina se mantiene el paralelismo, realizando cada iteración en dos ciclos, uno de espera. El programa de trayectoria desarrollado para ejecutarse en el simulador por usuarios poco expertos, realiza toda la secuencia de manera semiautomática, montando todo lo necesario y pasando de la máquina uno a la dos, a la tres y concluyendo. Pero resulta poco correcto esto de montar dos máquinas iguales de multiplicador doble, para calcular dos productos.

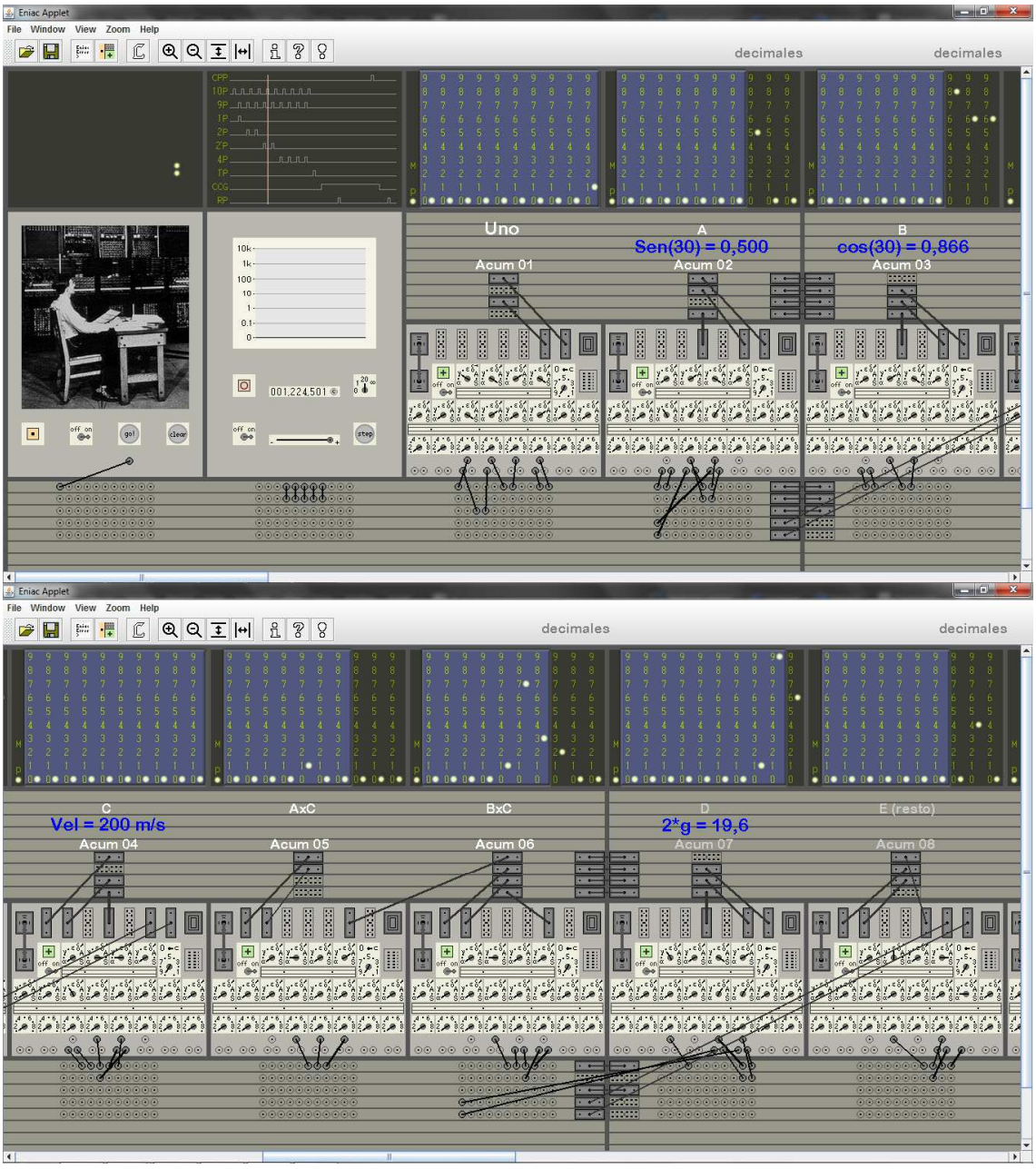


Imagen 70 – Trayectoria con el ENIAC primera parte, finalización

El ENIAC original al margen de los acumuladores, solo tenía dos máquinas, la del multiplicador y la del divisor que podía hacer también raíces cuadradas. Con ellas resolvía y encadenaba todos los cálculos, porque tenía mecanismos de

entrada y salida a memoria. Enviaba los resultados a memoria, imprimiéndolos en tarjetas perforadas, además era manejado en modo manual por usuarias expertas y podía recuperar los datos de memoria para reutilizarlos en el segundo producto, sin necesidad de montar dos máquinas diferentes. Por otro lado, en el simulador solo se dispone de una instrucción de bifurcación condicional, o *if* que resulta aparatosa de montar; en el ENIAC tenían además el programador maestro que para bucles fijos era muy útil.

El programa implementado en las imágenes 68 y 69 tiene el siguiente código estructurado en las rutinas de las tres máquinas y las tres de enlaces que realizan la comunicación, pase de datos entre máquinas y su activación. Los datos de entrada se colocan en los acumuladores dos, *sen(a),* tres, *cos(a)*, cuatro, *velocidad*, y siete, *2*g*. Y ejecutados los cálculos, aparecen los resultados en los acumuladores catorce, el *tiempo*, quince, *altura* y dieciséis, *alcance*.

```
/** Máquina 1 Calcula el producto de A y B por C */
                              //Fase de carga, todos los datos
  Acum1= 1, Acum2= A;              //Comienzo, pulso A-línea 1
  Acum3= B;                        //Pulso A-línea 2
  Acum4= C;                        //Pulso A-línea 3
  Acum7= D;                        //Pulso A-línea 4
                              //Fase de ejecución cíclica
  if(Acum4 >= 0){                  //Pulso A-línea 5
                                   //Espera
  Acum5+= Acum2,                   //P A-línea 6 y p A-línea 5
  Acum6+= Acum3, Acum4-= Acum1;
                              //Fase de restauración
  }                                //Espera
  Acum5-= Acum2,                   //Pulso A-línea 7, enlace 1
  Acum6-= Acum3, Acum4+= Acum1;


/** Enlace 1 Envía los productos A*C y B*C */
                              //Fase entre máquinas
  Acum11= Acum8= Acum5,          //Pulso A-línea 8, máquina 2
  Acum10= Acum6;


/** Máquina 2 Calcula cociente y resto de E dividido por D */
                           //Fase de carga de datos, ya lo están
                              //Fase de ejecución cíclica
  if(Acum8 >= 0){                 //Pulso A-línea 9
                                  //Espera
  Acum8-= Acum7, Acum9+= Acum1;  //P A-línea 10 y p A-línea 9
```

```
                                  //Fase de restauración
  }                                   //Espera
  Acum8+= Acum7,  Acum9-= Acum1; //Pulso A-línea 11, enlace 2


/** Enlace 2 Envía el cociente E/D */
                                  //Fase entre máquinas
  Acum14= 4*Acum9, Acum12= Acum9; //Puls C-línea 1, máquina 3


/** Máquina 3 Calcula el producto de F y G  por H */
                              //Fase de carga de datos, ya lo están
                                  //Fase de ejecución cíclica
  if(Acum12 >= 0){                //Pulso C-línea 2
                                   //Espera
  Acum13+= Acum10,                 //P C-línea 3 y p C-línea 2
  Acum15+= Acum11, Acum12-= Acum1;
                                  //Fase de restauración
  }                                   //Espera
  Acum13-= Acum10,                 //Pulso C-línea 4, enlace 3
  Acum15-=Acum11, Acum12+= Acum1;


/** Enlace 3 Envía el producto F*H */
                                  // Fase entre máquinas
  Acum16= 4*Acum13;                //Pulso C-línea 5, finalizar
```

En el programa trayectoria como en el resto, se pueden modificar fácilmente los datos iniciales del transmisor de constantes para efectuar nuevos cálculos, sin más que pulsar luego los botones *clear* y *go!*. El coste o tiempo de ejecución de este programa oscila porque depende de tres operaciones y de las relaciones entre sus factores. En una estimación y con estos decimales, se puede considerar que es algo mayor que seis por la talla de la velocidad en metros. Las operaciones de carga iniciales y la restauración se pueden ignorar.

Con esto se cierra el programa trayectoria que ha realizado prácticamente todo el proceso desde el problema físico con sus fórmulas y datos iniciales, a la implementación en el ENIAC de un programa para resolver los cálculos de una trayectoria, proporcionando los resultados finales. Estos programas demuestran que el ENIAC era algo más que una calculadora electrónica de válvulas de vacío y sirven para hacerse una idea de sus capacidades, que hoy nos pueden parecer muy limitadas, pero en su tiempo marcó un hito en el desarrollo de los computadores modernos y fue el ensayo para definir la arquitectura.

Sobre la controversia de quien fue el primer ordenador no se entra, pero las fechas están muy claras y el ENIAC no es el primero; lo que no está tan claro y es donde se debería centrar el debate, es sobre que se considera un computador como tal y ahí si se opina. Visto lo sucedido en esos años, se considera que el Z3 y Z4 ya eran computadores, el *ABC* parece un eslabón entre las calculadoras y los computadores que no acababa de funcionar correctamente, aunque legalmente una sentencia diga que fue el primero y los Colossus fueron tan secretos que no se tienen datos para opinar.

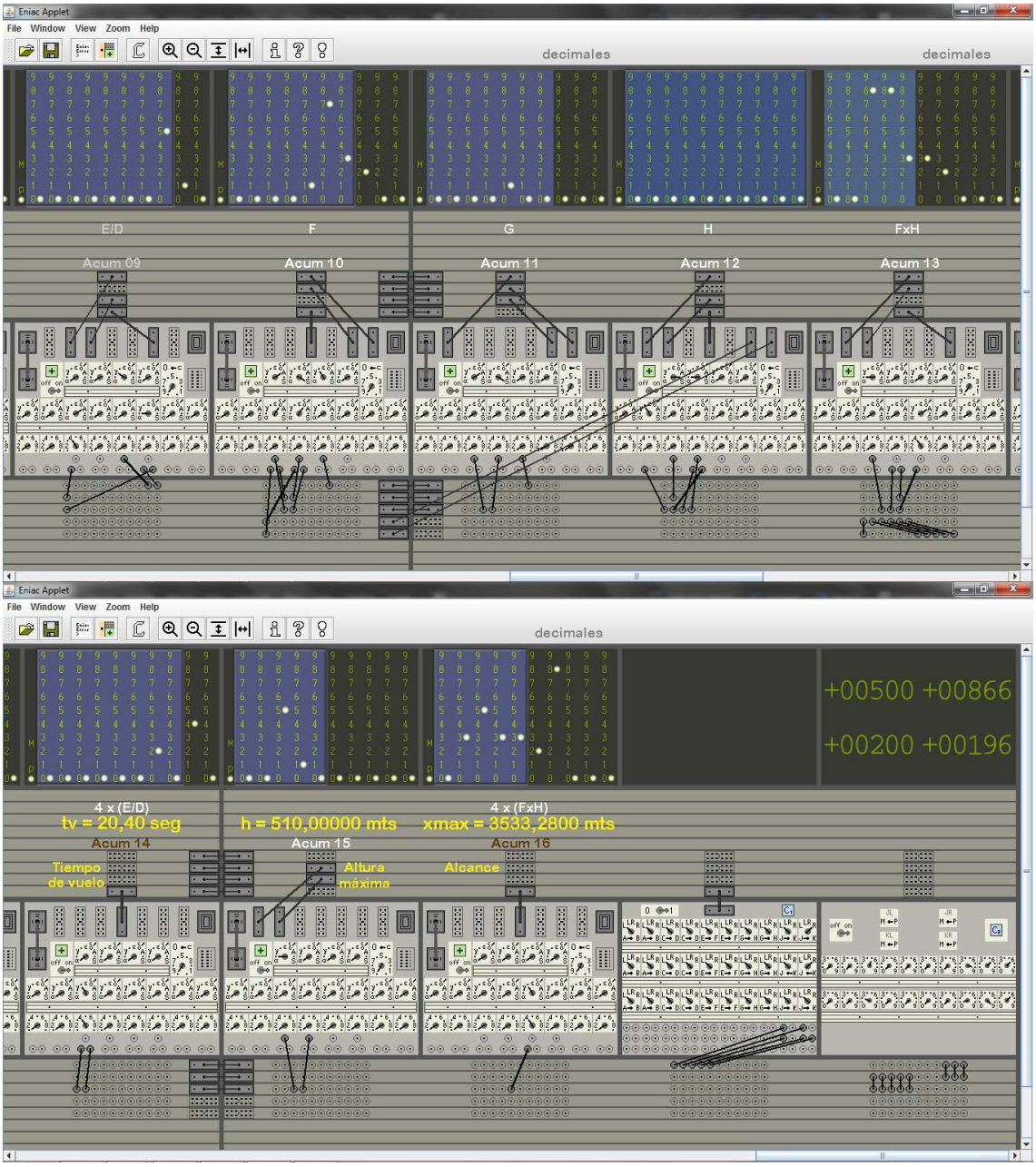


Imagen 71 – Trayectoria con el ENIAC segunda parte, finalización

6. Página multimedia

La página multimedia sale un poco fuera del proyecto y necesita coordinarse con el museo de la informática, así aunque se ha confeccionado una cuyo aspecto se muestra en las imágenes 72, 73 y 74, realmente solo se proporciona el contenido divulgativo sobre el ENIAC, pues ellos controlan el estilo, mantenimiento y demás tareas propias del portal, en pocas palabras el continente. Por otro lado la información del proyecto ha profundizado mucho en el diseño, funcionamiento y programación del ENIAC, lo cual se aleja de la idea inicial con un tema ligero, de perfil medio para usuarios noveles. Ello no es malo, pero deja la página multimedia divulgativa algo descolgada en relación al trabajo, que con este contenido y programas parecen más adecuados para una página de usuarios que quieran profundizar en computadores históricos.

Imagen 72 – Página multimedia

Consultado el tutor sobre el enfoque de la página, se mantiene el objetivo inicial que busca un tipo informativo con su público, por lo que se realiza un boceto de la página multimedia, siguiendo las directrices y dejando el proceso abierto con el museo, para realizar una versión definitiva que contemple los dos aspectos. Se toma como modelo la página del museo para estructurar la información del ENIAC, desarrollando una página multimedia local con información e imágenes, aunque la definitiva se ajustará a las prescripciones oficiales, tanto en contenido como extensión o recursos.

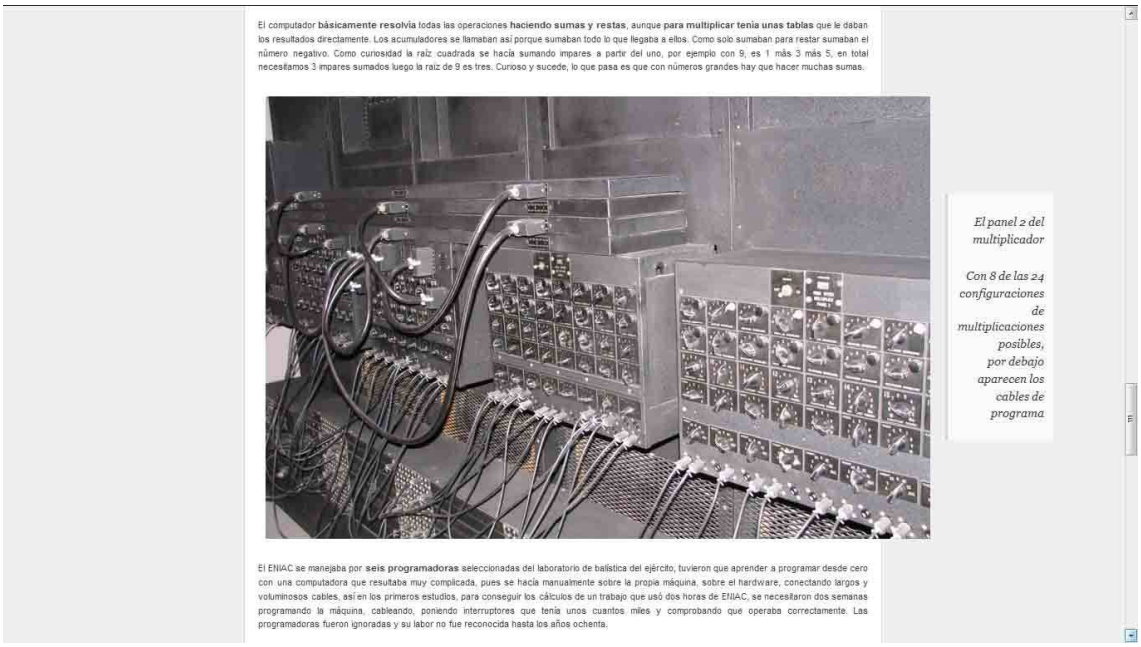


Imagen 73 – Página multimedia

Y en este apartado no hay mucho más que exponer, reiterando que se ha realizado una página multimedia con el formato del museo, empleada en local para obtener estas imágenes, estando pendiente de concretar el contenido propuesto y la subida, con los encargados oficiales de la misma.

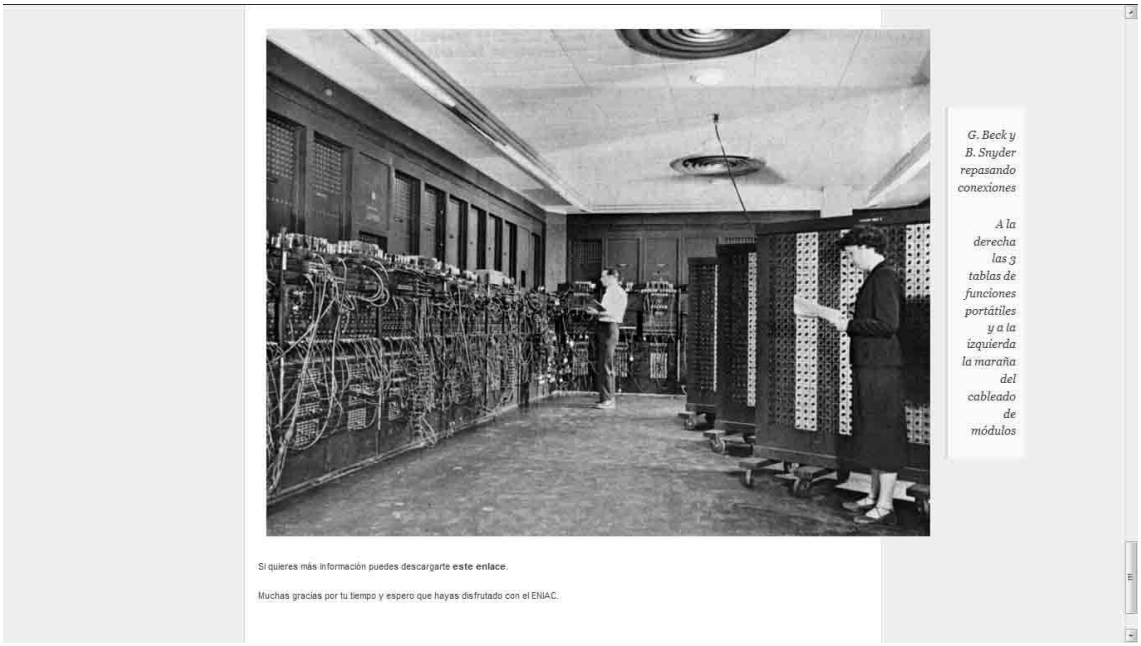


Imagen 74– Página multimedia

7. Conclusiones

Una vez acabado el proyecto se repasa el contenido y los objetivos propuestos al comienzo, que eran realizar un trabajo de investigación y documentación sobre el diseño y funcionamiento del computador ENIAC, desarrollando alguna aplicación para el mismo, con la idea de lograr un documento divulgativo medio, dirigido a un público poco experto, que se pueda plasmar en una página multimedia para su difusión general desde el museo de la informática, aprovechando los medios en la red.

Sin embargo el desarrollo del proyecto ha profundizado mucho en el diseño y funcionamiento del ENIAC. Usando imágenes y los planos originales se ha realizado una extensa descripción gráfica del computador desaparecido hace sesenta años, que además se ha relacionado con el simulador eniac.jar que lo mantiene vivo actualmente, también se han diseñado nueve programas que recuperan de manera limitada las prestaciones iniciales del ENIAC, incluso se ha elaborado alguno, como la trayectoria que recoge el complejo proceso de programación en la época, para llegar a los cálculos desde las formulas físicas, aspectos que normalmente ya no se usan.

En el trabajo se ha seguido un método exhaustivo, muy apoyado en la descripción gráfica que surge de las imágenes, tratando de hacer un enfoque distinto, que se ha completado con un desarrollo bastante extenso de la programación para el ENIAC, donde al principio, resultaba extraño comprobar los pocos programas disponibles, achacado ello al poco interés que despierta la programación de computadores históricos. Sin embargo, el vacío inicial ha permitido en cierto modo, una amplia autonomía para organizarlo como un todo sin tener que estar pendiente de lo que otros han hecho.

Por ello se considera que los objetivos de investigar y documentar el ENIAC con su programación se han logrado con holgura, así como la página multimedia que se realiza siguiendo el planteamiento inicial. Con independencia del conocimiento y experiencia adquiridos, el proyecto ha resultado muy interesante pues ha requerido un proceso de investigación previa, más próximo a la historia que a la informática, para conseguir primero situarse en el escenario social y el contexto tecnológico de los años cuarenta, que había alrededor del ENIAC y ha resultado curioso ver los personajes aparecidos.

La documentación bibliográfica y las búsquedas por internet se han utilizado de manera general para aprender las cosas, pero prácticamente no se ha seguido ningún autor concreto de los que figuran en la bibliografía, sino más bien se han ido sumando los conceptos y las discrepancias de unos y otros, para formar un criterio propio que, a partir de ese punto, se ha plasmado en este trabajo. Sin embargo, se ha de señalar que se estudió con detalle toda la información en

torno a Till Zoppke y Raul Rojas que desarrollaron el simulador y el segundo además, ha escrito sobre computadores históricos.

También se ha intentado realizar un trabajo diferente al que ya circula por las bibliotecas o la red. El resultado de este proyecto puede servir de apoyo en estudios posteriores sobre el ENIAC, pues aunque no se ha profundizado en la tecnología del computador, bastante superada, se ha insistido bastante en su funcionamiento de manera global y tratando las relaciones entre elementos. Por otra parte ha resultado muy clarificador en ocasiones la propia patente, documento básico de conocimiento primitivo, que da la sensación no se aprovecha con toda la información que contiene y las dudas que puede resolver buscando directamente en sus esquemas y anotaciones.

Dos ideas han primado en esta memoria a nivel personal, por un lado intentar trasmitir la información de manera gráfica, vía que suele desaprovecharse en ocasiones, dejando las imágenes en meros adornos que sirven para que respire el texto. Y por otra, dar a la memoria un contenido que no está muy visto, con un enfoque que relaciona el ordenador viejo y su simulador por medio de la patente usada como nexo. Además una vez que la programación del ENIAC se dominó suficiente, se intentó desarrollarla al máximo para dejar ese apartado la más completo posible.

La sensación después de terminar, es de satisfacción por haber superado al computador que inicialmente era un desconocido de incierto porvenir. Ha faltado un poco de tiempo o tal vez ha sido demasiado extenso el campo abarcado, lo que sucede es que al principio el ángulo era reducido, pero según se fue adentrando, se empezaron a abrir múltiples vías que se han intentado seguir todas, pues lo que dejas atrás no se suele recuperar.

Cuando se presentó el tema se hablaba bastante de las chicas del ENIAC, las programadoras olvidadas o sacadas de la foto, y otra serie de aspectos conocidos que se han constatado y son reales. Eso llevó en el trabajo de investigar por investigar, a intentar identificar los actores, cosa que no ha costado demasiado trabajo en los hombres. Pero ha llamado mucho la atención el trabajo que ha costado identificar a las mujeres, lo estaba viendo aunque no era consciente, sencillamente porque suelen estar de espaldas, parecen de cara a la pared en un porcentaje que supera ampliamente la casualidad.

8. Bibliografía

Augarten, S. (1984). *Bit by bit: an illustrated history of computers*. Londres: George Allen & Unwin.

Barceló, M. (2008). *Una història de la informàtica*. Barcelona: Editorial UOC.

Brainerd, J. G. y Sharpless, T. K. (junio, 1999). The ENIAC. Proceedings of the IEEE, *87*(6), 1031–1041. Reimpreso de *Electrical Engineering*, *67*(2), 163–172, (febrero, 1948).

Breton, P. (1989). *Historia y crítica de la informática*. Madrid: Cátedra.

Campbell-Kelly, M. y Aspray, W. (2004). *Computer: a history of the information machine*. Westview Press, segunda edición.

Ceruzzi, P. (2003). *A history of modern computing*. Londres: MIT Press, segunda edición.

Coello Coello, C. (2003). *Breve historia de la computación y sus pioneros*. México: Fondo de Cultura Económica.

Eckert, J. Presper y Mauchly, John W. (1964). Patente del Electronic Numerical Integrator and Computer US3120606 A. Consultado el 2 de julio de 2014 en http://www.google.com/patents/US3120606

Fritz, W. Barkley. (1994). ENIAC – a problem solver. *IEEE Annals of the History of Computing*, *16*(1), 25–45.

Goldstine, H. (1980). *The computer from Pascal to von Neumann*. Princeton: Princeton University Press.

Hally, M. (2005). *Electronic brains: stories from the dawn of the computer age*. Londres: Granta Books.

Ifrah, G. (2008). *Historia universal de las cifras*. Madrid: Espasa Calpe, S.A., sexta edición.

Light, Jennifer S. (julio, 1999). When computers were women. *Technology and Culture*, *40*(3), 455–483.

Martin, C. Dianne. (invierno, 1995/1996). ENIAC: press conference that shook the world. *IEEE Technology and Society Magazine*, *4*(14), 3–10.

McCartney, S. (1999). *ENIAC: The triumphs and tragedies of the world's first computer*. Nueva York: Walker and Company.

Molero, Xavier. (2013). ENIAC: una máquina y un tiempo por redescubrir. *XIX Jornadas sobre la Enseñanza Universitaria en Informática* (pp. 241-248). Castellón de la Plana.

Rojas, R y Hashagen, U. (2000). *The First Computers-History and Architectures*. Londres: The MIT Press.

Schapranow, M. (2006). Eniac tutorial – the modulo function. Consultado el 2 de julio de 2014 en http://placebo.hpi.uni-potsdam.de/webhome/matthieu.schapranow/eniac/modulo/

Shurkin, J. (1996). *Engines of the mind: the evolution of the computer from mainframes to microprocessors*. Nueva York: W. W. Norton & Company.

Swedin, E. y Ferro, D. (2005). *Computers: the life story of a technology*. Baltimore: The Johns Hopkins University Press.

War Department. (1946). Comunicado de prensa del Departamento de la Guerra, emitido el 15 de febrero. Consultado el 2 de julio de 2014 en http://americanhistory.si.edu/comphist/pr1.pdf

Williams, M. (1997). A history of computing technology. IEEE Society Press, Los Alamitos, CA, segunda edición.

Zoppke, T. (2004). Simulating the ENIAC as a Java Applet. Berlín. Consultado el 2 de julio de 2014 en http://kinnla.github.io/eniac/doc/eniac_simulation_thesis.pdf

Zoppke, Till y Rojas, Raul. (abril, 2006). The virtual life of ENIAC: simulating the operation of the first electronic computer. *IEEE Annals of the History of Computing*, (28), 18–25.

Zoppke, T. (20013). ENIAC simulation. Aplicación simulador eniac.jar. Consultado el 2 de julio de 2014 en http://kinnla.github.io/eniac/