Universidad Nacional Autónoma de México

Facultad de Ingeniería

Sistemas Operativos

Grupo 06

Semestre 2026-1

- Cano Vázquez Axel Zaid: 320198687
- Tejeda Vaca Abraham: 321328456

Proyecto 1 – MiComputer: Fascículo 10

TI-99/4A

Lo que más me sorprendió del fascículo 10 fue que la TI-99/4A usara un procesador de 16 bits (TMS9900) cuando muchas máquinas domésticas aún eran de 8 bits. Eso cambia la forma en que la computadora organiza su memoria y "dialoga" con los periféricos. También me llamó la atención el ecosistema: cartuchos ROM para ampliar el BASIC, caja de expansión para agregar controladora de disco o RS-232, video con 16 colores y sonido de 3 voces.
Leyéndolo con los lentes de Sistemas Operativos, vi varias conexiones claras:

- Dispositivos y E/S. Aunque no haya un "SO" moderno, el sistema depende de rutinas (firmware) para cassette, joystick, video y serial. Es justo lo que vemos en clase: alguien tiene que abstraer el hardware con reglas y tiempos (drivers o servicios básicos).

- Memoria limitada → decisiones de diseño. Con 16 KB para el usuario, mucho recae en ROM: el BASIC (y el ampliado) aporta funciones "de sistema" (gráficos tipo sprite, voz) sin mover todo el software. Es una lección de cómo la API del sistema crece por módulos sin romper lo demás.

- Camino de ejecución. El BASIC interpretado es cómodo para probar rápido, pero es más lento. Si quieres velocidad real (juegos/gestión), pasas a código máquina o a un lenguaje compilado. Es el trade-off clásico entre tiempo de desarrollo y tiempo de ejecución que vemos cuando analizamos costos de CPU y memoria

Además, es difícil de imaginar que la única forma de gestionar software fuera exclusivamente por medio de cartuchos ya sea para mejorar las funcionalidades del ordenador mediante el uso de un cartucho BASIC "ampliado", integrar un nuevo lenguaje de programación que se adecue a la tarea u otro motivo que se tenga en mente, pero gracias a este artículo podemos percibir el gran crecimiento exponencial que han tenido las computadoras personales a lo largo del tiempo y como se tenía en cuenta la modularidad del sistema ofreciendo soluciones particulares a las limitaciones que tenía el ordenador con las numerosas ampliaciones de software y hardware con el que contaba.

Reflejándose en este caso en las soluciones que se daban a cierto requerimiento ajustado al contexto y el desarrollo del hardware del momento, por ejemplo, como lo marca el artículo sobre el uso de una gran cantidad de componentes discretos que requería el ordenador para funcionar, usando resistencia y transistores que actualmente pudieran ser fácilmente reemplazados por un chip; también, como en el caso de uso de la "Caja de ampliación periférica" que a pesar de ayudar a superar las limites respecto al hardware con las que contaba el TI-99/4 por ejemplo al

expandir la cantidad de RAM era necesario esta caja, pero al comparar esta solución con las opciones de la actualidad puede resolverse inmediatamente reemplazando la memoria RAM por una de mayor capacidad o agregar otra en dado caso que hubiera en un slot interno disponible.

Intérpretes vs. compiladores

Este texto me sirvió para ordenar una confusión común: intérprete y compilador no hacen lo mismo ni cuestan lo mismo. Con un intérprete (como BASIC en muchas micros), escribes, das RUN y el sistema traduce línea por línea cada vez. Es barato en memoria y muy cómodo para aprender o depurar, pero repite trabajo y corre más lento. En cambio, con un compilador, primero generas un archivo objeto (código máquina) y luego lo ejecutas; el costo está antes (configurar editor, compilar, corregir errores), pero el programa corre mucho más rápido.

Lo ligué con lo que hemos visto en el curso y en prácticas: cuando trabajas con ejecutables/objetos, cambia la forma en que el sistema carga en memoria, ubica símbolos, y cómo interactúas con dispositivos. También entendí mejor por qué algunos paquetes de la época venían en cartuchos ROM: además de rendimiento y comodidad, sirven como “protección” (no se modifican fácilmente), aunque te amarran al proveedor.

Entre la TI-99/4A y el tema de intérpretes vs. compiladores, me quedo con dos ideas que encajan perfecto con la materia: 1) la importancia de abstraer bien el hardware (porque tu experiencia depende de esas rutinas y de cómo se organiza la memoria), y 2) que las decisiones de ejecución (interpretar, compilar, cartucho) son decisiones de sistema, no solo de quien programa. Eso explica por qué, incluso con máquinas “modestas”, se podían lograr experiencias muy diferentes según cómo estuviera armado el software y el soporte del sistema.

Síntesis de voz

Esta lectura nos indica sobre dos maneras de poder capturar y procesar la voz, donde la primera se enfoca en una solución de software encargada de guardar en bloques de información “fonemas” cada letra que contiene una palabra mediante el análisis de frecuencias que componen a la voz humana creando desde cero el sonido de la letra y combinando los fonemas poder construir cualquier palabra alojándose dichos bloques en órdenes de BASIC llevándose la mayor carga de trabajo la parte del software que del hardware, sin embargo, este procedimiento quita las características de una voz humana manteniendo nada más la claridad en la pronunciación.

Por otro lado, el segundo método requiere de más hardware, pero debido a esto se mantiene las características de una voz humana por el equipo dedicado con el que cuenta como memoria, convertidores analógico-digital y viceversa, entre otros; los cuales los primeros captan la señal para procesarla en información digital para así comprimir y guardar los datos en una memoria ROM. Para replicar la pronunciación de cualquier palabra nada más se tiene que brindar su dirección en la ROM para recuperar la información digital y mediante un convertidor digital-analógico transformarlo en sonido.

Siendo una parte importante el uso de la memoria para el manejo de la información, uso de discos para almacenar los datos para después ser extraídos por programa especializado, interrupciones que guían a rutinas de manejo por eventos exteriores como la información de un dispositivo de entrada y su mapeo en memoria para la gestión de dispositivos de entrada/salida.

Lo importante aquí es las distintas aplicaciones en las que se puede emplear este proceso, que como menciona el fascículo son demasiados, destacando su uso en los avisos grabados de las distintas formas de transporte que hay, como avión, ferrocarril y otras terminales; como parte del tablero de un automóvil que ayuda al conductor a saber sobre el estado del automóvil sin apartar la vista del camino, también para las computadoras personales en el apartado de los videojuegos para avisar de ataques de enemigos o la puntuación que tiene y así el jugador se concentre en su totalidad, entre otros.

Referencias:

MiComputer. (1984). Delta. Fascículo 10. Recuperado el 16/08/2025 de: http://web8bits.com/Coleccion/Libros/Graficos/MiComputer/pdf/Fasciculo_010.pdf