Instituto Tecnológico de Costa Rica

Escuela de Computación

IC5701 Compiladores e Intérpretes

Apuntes de clase
Fecha: 15 de febrero del 2017

Realizado por:
Adrián Jesús Álvarez Calderón
2014071059

Profesor:
Francisco José Torres Rojas

I Semestre 2017

Tabla de contenido

Quiz 1

1. Defina detalladamente los siguientes conceptos:
    a. ENIAC
    b. John Backus
    c. “First Draft”
    d. Maurice Wilkes
    e. Ensamblador
    f. Grace Murray Hopper
    g. Microprogramación
    h. Productividad
    i. John Eckert
    j. Arquitectura

Clase
(Continuación de los lenguajes de alto nivel)

Lenguajes de Alto Nivel:

Los lenguajes de programación vienen a posicionar una nueva capa en una jerarquía que tiene el propósito de hacer funcionar un hardware para lograr un objetivo o resultado en particular. Además, se mantiene la relación productividad/eficiencia al igual que en cada una de las capas inferiores como lenguaje ensamblador, lenguaje máquina y microprogramación. Siendo esta relación, respecto a sus inferiores, mayor productividad (máxima) pero castigando un poco más la eficiencia.

Con llegada de lenguajes de alto nivel se empieza a tener una independencia del hardware, por lo que se omiten preocupaciones respecto a si un código (en alto nivel) va a funcionar en otro equipo con características muy particulares como un

procesador en específico con cierta cantidad de bits, un tamaño de palabra en particular, etc.

Es decir, se logra tener códigos y programas portables, algo que marca una importante novedad en el momento de su desarrollo, esto proporciona una abstracción para el programador por detalles específicos de arquitectura. Ya los desarrolladores pensaban en un lenguaje como FORTRAN, lenguaje de mucho más acceso que otro de niveles inferiores (jerarquía) como lenguaje ensamblador o lenguaje máquina, por lo que finalmente el hardware se vuelve completamente irrelevante en el desarrollo.

Esto era posible ya que el lenguaje de alto nivel era el mismo para cualquier equipo en el cual se debía ejecutar, no obstante, la variación se da en el compilador de este lenguaje de alto nivel que probablemente hacía la traducción específica al lenguaje máquina de diferentes computadoras y siendo estos resultados (y compiladores) diferentes. Por lo que el esfuerzo de desarrollar el compilador para una máquina en específica valía la pena para que en un nivel superior cualquier código fuera compatible.

Máquinas:

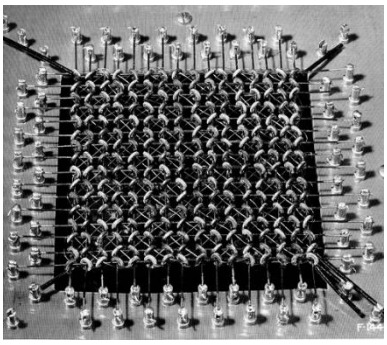
IBM 701:

- Considerada la primera computadora científica comercial.
- Desarrollada en abril de 1952.
- IBM dominaba el mercado computacional al alquilar sus computadoras.
- Se fabricaron 19 equipos.
- Configuración:
  - 2048 palabras de 36 bits aumentable a 4096 palabras de 36 bits
  - 150 mil instrucciones por segundo
  - 150 tarjetas leídas por segundo
  - 100 tarjetas perforadas por segundo

- Las instrucciones eran de 18 bits
  - 1 bit signo
  - 5 bits de operación (32 instrucciones)
  - 12 bits de dirección (4096 direcciones de media palabra)
- Sumaba, restaba, multiplicaba y dividía.
- El costo de alquiler era de $23.750 al mes de 1952, que corresponden a $219.970 en el 2017.
- Cada componente o consumible (como tarjetas perforadas) de la computadora era marca IBM. El mantenimiento y la capacitación también corría por cuenta del servicio de IBM.
- MTBF: 30 minutos

IBM 704:

- Primera computadora con punto flotante.
- Desarrollada en 1954 y utilizada hasta 1960.
- IBM vendía o alquilaba 200 máquinas.
- El costo el alquiler era de $35.550 al mes en 1954, que corresponde a $319.272 en el 2017.
- Doble de rápida que la IBM 701 pero completamente incompatible.
- Novedad: Dispositivos para copiar tarjetas perforadas a una cinta magnética de manera off-line (es decir, fuera de la línea de ejecución del equipo).
- Contaba con memoria de núcleos de ferrita.
- Las instrucciones eran de 36 bits.
- MTBF: 8 horas
- Se disponía de FORTRAN para su compilación, su producto estrella que impulsa a la atracción de la computadora.



Memoria de núcleos de ferrita

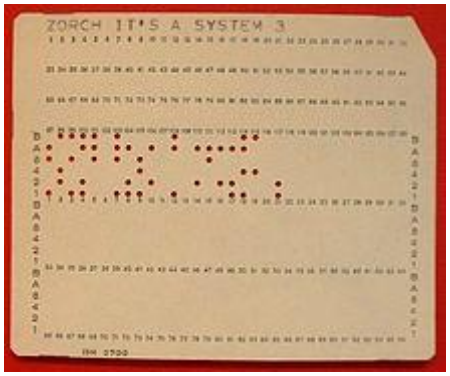
Máquinas dedicadas:

Manera en que operaban computadoras como IBM 701 e IBM 704.

En el alquiler de estas computadoras se disponía de la reservación de horas para su utilización, puesto que había una cantidad disponible relativamente pequeña. Por lo general se ubicaban en ciertos lugares especiales como universidades o ciertas industrias de investigación. Este modelo era muy común en los años 50. En general la persona que utilizaba la computadora era el programador, el científico y el ingeniero responsable de hacer todo. La secuencia de boot era completamente manual, lenta y complicada, además era propensa a errores. Aproximadamente duraba 5 minutos el boot en el caso ideal que raramente se daba.

Uso típico de la máquina (caso ideal):

1. Configuración de todos los tableros de control.
2. Colocar FORTRAN en la lectora de tarjetas. Dar boot al computador para que lea el FORTRAN y sepa cómo compilarlo.
3. Colocar la fuente del programa que se quiere compilar, escrito en FORTRAN, en la lectora de tarjetas. Se intentará compilarlo, esto puede requerir una o varias pasadas (lectura del fuente).
4. Si no hay errores, el compilador perfora el programa equivalente al ensamblador, es decir, realiza el proceso de compilación dejando como residuo un programa en ensamblador en tarjetas perforadas.



5. Colocar el Ensamblador en la lectora de tarjetas. Se da boot al computador para que pueda ensamblar un fuente.
6. Colocar el fuente ensamblador (producto de la compilación de FORTRAN) en la lectora de tarjetas. Se intentará ensamblar, esto puede requerir una o varias pasadas (lectura del fuente).
7. El ensamblador perfora el programa equivalente en lenguaje máquina.
8. Colocar lenguaje máquina, obtenido en el paso anterior, en la lectora de tarjetas. Dar boot al computador para cargar dicho programa.
9. Recolectar el listado de la impresora (resultados del programa).

10. Go to 1 para un nuevo programa.

Las personas que utilizaban la computadora usaban gabacha *porque era cool*. Era posible que en alguna etapa intermedia se obtuviera un listado de errores de compilación.

Analizando la Máquina Dedicada:

Ventajas:

- La principal ventaja era que "*era cool*", pues para la época la computación era algo muy avanzado.
- Todos los procesos los hacía la misma máquina.
- El programador tiene un control absoluto de la máquina.
- Era suficiente para la época, pues eran poco expertos al inicio.
- Fácil establecer responsabilidades y controles.

Desventajas:

- Mucho trabajo repetido entre un usuario y otro por la carga del compilador de FORTRAN, el ensamblador, etc, trabajado desperdiciado.
- No había estándares.
- Los programadores debían ser expertos en hardware.
- Se da un uso muy ineficiente de los recursos.

Mejorando la Máquina Dedicada:

Se llega a una idea administrativa en la que se evidencia que hay mucho desperdicio de recursos y otras desventajas debido a la manera de operación de la computadora. Por lo que se crea el rol de Operador de Computadoras: ¡Fuera Programadores! De manera que va a existir una persona especialista en que la máquina funcione de una mejor manera. Se reciben las tarjetas perforadas de muchas personas y la tarea del operador es empezar primero a compilar todos los programas a ensamblador, posteriormente se encargaba de ensamblar todos los programas, para finalmente ejecutarlos y completar las tareas necesarias para el cliente que llevó sus tarjetas. Esto presenta muchas ventajas al tener que cargar una sola vez el compilador y una sola vez el ensamblador. Ahora se tiene una

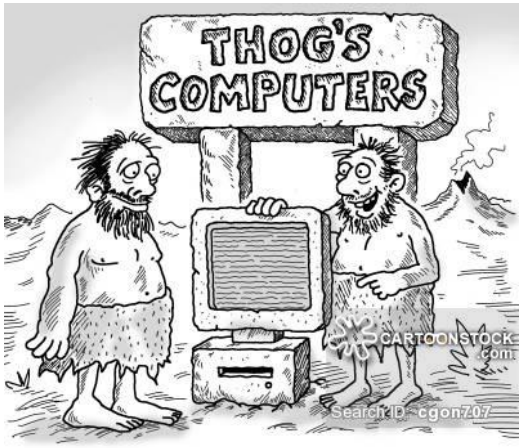
configuración estándar del tablero para así hacer los trabajos más rápidos y que sean más baratos. A esto se le llamaba procesamiento en lotes (Batch Processing). Se da un incremento en el throughtput (cantidad de unidades terminadas por unidad de tiempo) en el centro de cómputo. La productividad mejoró considerablemente en el Procesamiento Bach respecto a la Máquina Dedicada.

Los operadores funcionaban muy bien, pero se pensaba que el trabajo era muy mecánico, tal vez se podían llegar a quitar los operadores. Por lo que se decide crear un software que lo reemplace.

> "*La historia de la computación es la historia de la pereza, pero también es la historia de la avaricia*" (Torres, 2017)

Sistemas operativos Primitivos:

Contextualizando históricamente, existen ya ciertos lenguajes de programación como FORTRAN, COBOL, ALGOL, muchos compiladores, hay más dispositivos modernos y fallan menos, unidades de disco primitivas, para algunas cosas ya no es necesario de hacer con tarjetas perforadas sino con algún otro mecanismo, como que un compilador esté en disco o en una cinta magnética, por ejemplo. Se crea el sistema operativo para deshacerse del operador. No obstante, aún los primeros sistemas operativos se requerían de tarjetas perforadas, en las que se encontraban el sistema operativo. Este tenía la posibilidad de correr programas de usuarios uno después del otro. El objetivo era que fuese completamente automático, es decir, después de recibir los programas se compilaban y ejecutaban uno tras otro automáticamente sin intervención o la mínima posible.



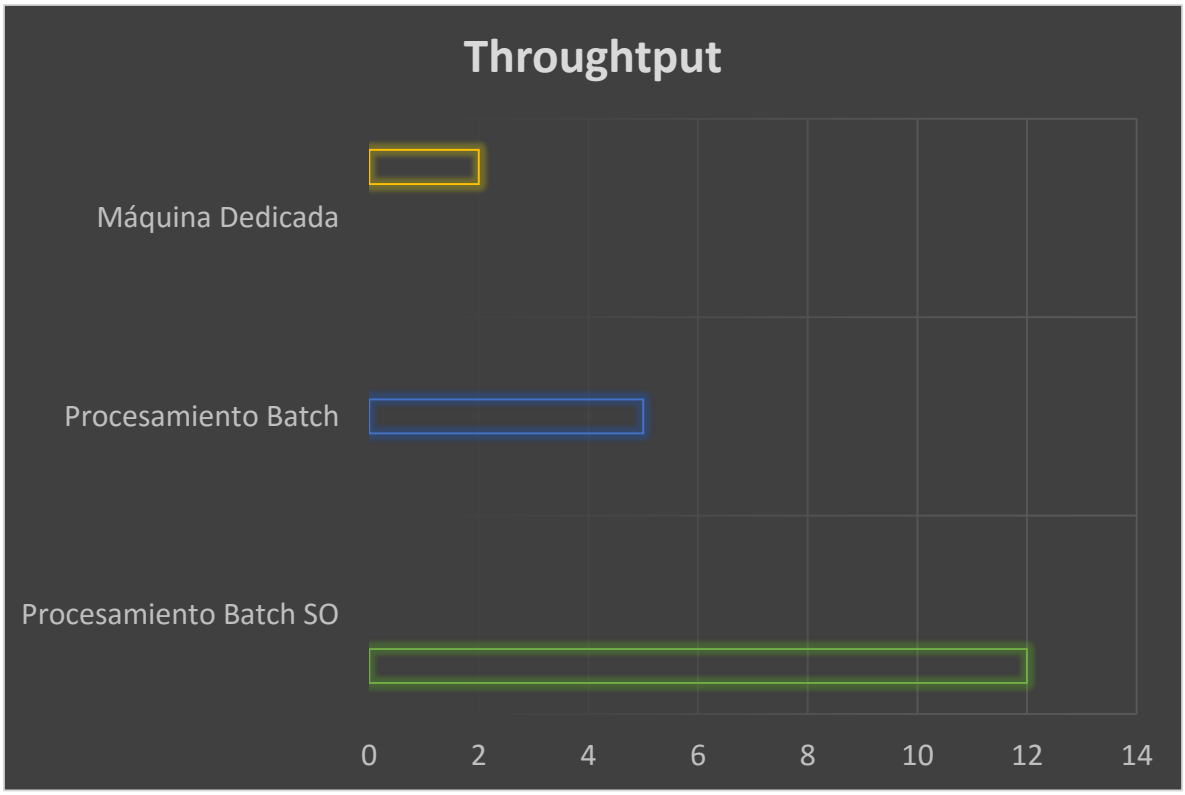
"... and this one comes with two trilobites of memory!"

Lo que planteaba ciertos problemas, como por ejemplo limitar cuando inicia un programa y cuando termina. Puede que estos programas eran en diferentes lenguajes de alto nivel, entonces era importante diferenciar el compilador requerido. Diferenciación entre las tarjetas de código fuente, y tarjetas de datos. Para ello se creó el JCL.

Job Control Language (JCL):

- Es la primera interfaz entre el usuario y el sistema operativo.
- Inicialmente era tarjetas intercaladas entre los trabajos de los usuarios.
- Tenía una sintaxis primitiva y difícil de entender.
- Es antecesor de Shell languages o Interfaces Gráficas.
- Eran muy específicos para cada sistema operativo.

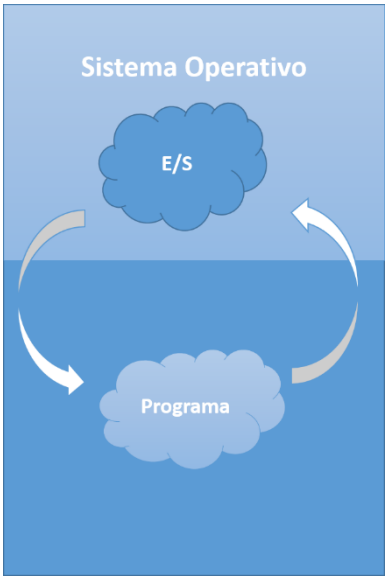
Productividad de centros de cómputo:



Entrada/Salida Primitiva:

- Arquitectura de von Neuman sugiere un espacio de direcciones Entrada/Salida. Es decir, hay instrucciones para este control.
- Instrucciones máquina para ese espacio.
- No era fácil y en dispositivos nuevos eran más complejos.
- Al inicio cada uno escribía sus rutinas es Entrada/Salida para utilizar los dispositivos.
- Se las compartían estas rutinas o instrucciones entre programadores.
- Se crean bibliotecas de Entrada/Salida agregadas a cada programa.

Todos los programas requieren de operaciones de entrada y salida, además la idea del sistema operativo en memoria funcionó de maravilla. Se propone que las funciones de entrada y salida estén en memoria, y que sean parte del sistema operativo.



De manera que cada vez que un programa ocupaba operaciones de entrada / salida, ya no debía de utilizar librerías, sino llamaba al sistema operativo para que se encargase de ello.

Servicios del Sistema operativo:

Las operaciones entre un programa y el sistema operativo, por ejemplo, ejecutar funciones de entrada/salida se realizan a través de servicios que proporciona el sistema operativo. Estas operaciones, el sistema operativo las realiza a través de un *Device Driver* (software especialista en controlar un dispositivo en particular). Dado que esto funcionó bien, se empezó a generalizar los servicios que puede dar un sistema operativo, cada vez dando más *system calls*, que son instrucciones virtuales.

Estos system calls son estrictamente específicos para cada sistema operativo. Por lo que con el paso del tiempo los sistemas operativos crecieron en tamaño y complejidad (tienen muchas piezas).

Definiciones:

- Real: Lo que se ve y existe.
- Virtual: Lo que se ve, pero no existe.
- Transparente: Lo que no se ve, pero existe.
- Máquina: Dispositivo (real, virtual o transparente) que analiza una tarea computacional.

El sistema operativo se vuelve dueño del hardware, pero a cambio de dar servicios.