TEC | Tecnológico de Costa Rica

Apuntes de la clase del 22 de Febrero del 2017

Apuntador: Ariel Montero Monestel

Curso: IC5701 Compiladores e Intérpretes-G40

Profesor: Dr Francisco J. Torres-Rojas

Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Sede San José
I Semestre, 2017

Quiz

1) Defina detalladamente los siguientes conceptos:
    a) Características del COBOL
    b) System Call
    c) Device Driver
    d) Máquina Multiniveles
2) Suponga que su sistema Linux tiene un compilador de C, versión 7. Ud recibe por correo los siguientes archivos:
    a) El programa fuente de la versión 10 del compilador de C, escrito en C
    b) El fuente de un compilador de Fortran que genera C, escrito en C
    c) Un compilador de Cobol, escrito en Fortran, que genera lenguaje máquina

    Usando diagramas T y explicaciones detalladas, diga lo mejor que se puede hacer con esto.

Comentarios y dudas previas al inicio de la clase

El profesor tocó el tema del ranking y dió una breve explicación de los siguientes pseudónimos que aparecen en el mismo:

- **Perfecto:** Es el estudiante imaginario que lleva 100 en todo. Lo ideal es parecerse lo más posible a él.

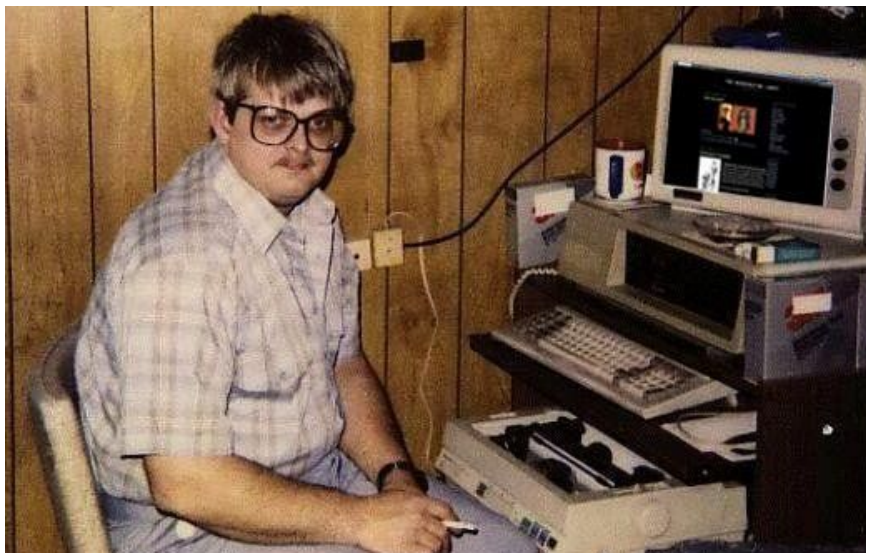


Figura 1: Mr. Perfecto.

- **Promedio:** Es el promedio de las notas de los estudiantes del grupo. Entre más arriba esté el promedio, mejor es el grupo, y viceversa. Lo ideal es estar por encima del promedio; aunque claro está que no todos pueden estarlo.

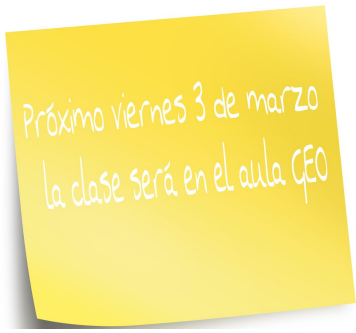
- **Pasando:** Es el estudiante que se saca 67.5 en todo. Pasará el curso con 67.5
- **Último:** Es el que realiza justo lo necesario para pasar, y es el último que va a pasar. Si usted está por debajo de *último* al final del curso, significa que perdió el curso.

Además de explicar estos pseudónimos, también explicó las columnas:

- **Falta:** indica cuántos puntos necesitamos para pasar el curso
- **Máximo:** Nota máxima a la que podemos aspirar si obtenemos 100 en todo lo que falta.
- **Proyección:** Tendencia de su nota.

Con respecto al proyecto:

- No es necesario hacer documentación
- Se debe traer ejemplos listos para correr



Niklaus Emil Wirth (1934 -)

- Winterthur Suiza. Científico de la computación.
- En 1959 obtiene el título de Ingeniero en Electrónica en la Escuela Politécnica Federal de Zúrich (ETH) en Suiza.
- Wirth fue el jefe de diseño de los lenguajes de programación Euler, Algol W, Pascal, Modula, Modula-2 y Oberon. Recibió el Premio Turing por el desarrollo de estos lenguajes de programación en 1984.
- Su libro "*Algoritmos + Estructuras de datos = Programas*", recibió un amplio reconocimiento, y aún hoy resulta útil en la enseñanza de la programación.

Lenguaje Pascal

- El objetivo del lenguaje era ser simple, elegante y apropiado para enseñar a programar con buenos hábitos.
- Programación estructurada.
- El lenguaje se volvió exitoso e influyente, tanto así que se empezó a usar alrededor del mundo de manera pedagógica

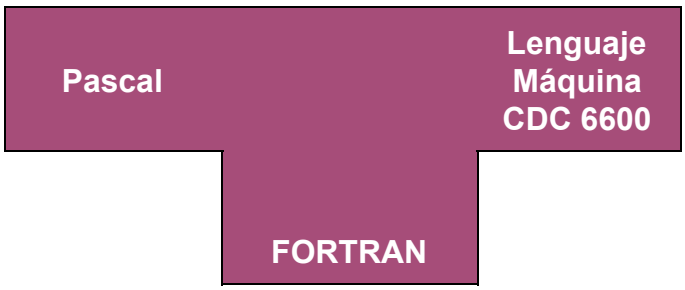
Bootstrapping

La idea del bootstrapping es simple: se escribe un compilador que recibe **A**, genera **B** y este compilador está escrito en **A**, y luego se pone al compilador a compilarse a sí mismo.

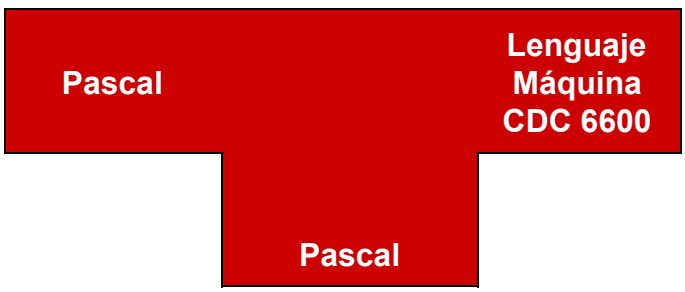
El resultado es un compilador de A a B escrito en B. Puede sonar un poco paradójico que el compilador se compile a sí mismo. En breve veremos cómo se resuelve esta aparente paradoja.

El bootstrapping de Pascal

- En un principio se buscaba desarrollar un compilador de Pascal en FORTRAN. El target sería el lenguaje máquina de la CDC 6600

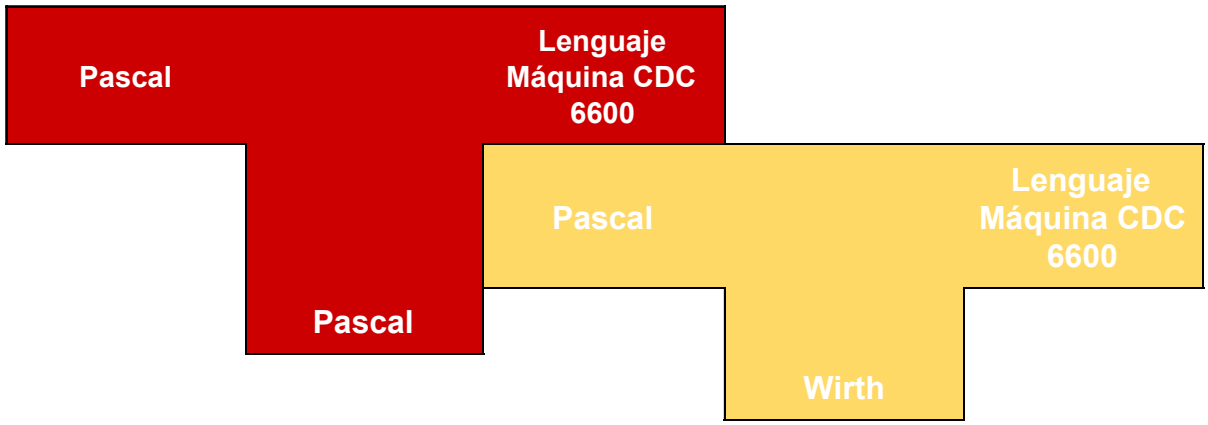


- A Wirth no le gustaba FORTRAN, y decide escribir el compilador de Pascal en Pascal.

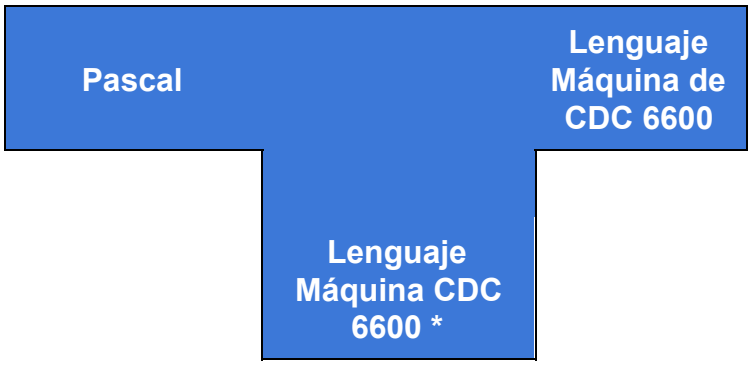


Aquí es donde surge la paradoja: **¿Cómo *&!*#$ compilo el compilador de Pascal cuyo Host está en Pascal?**

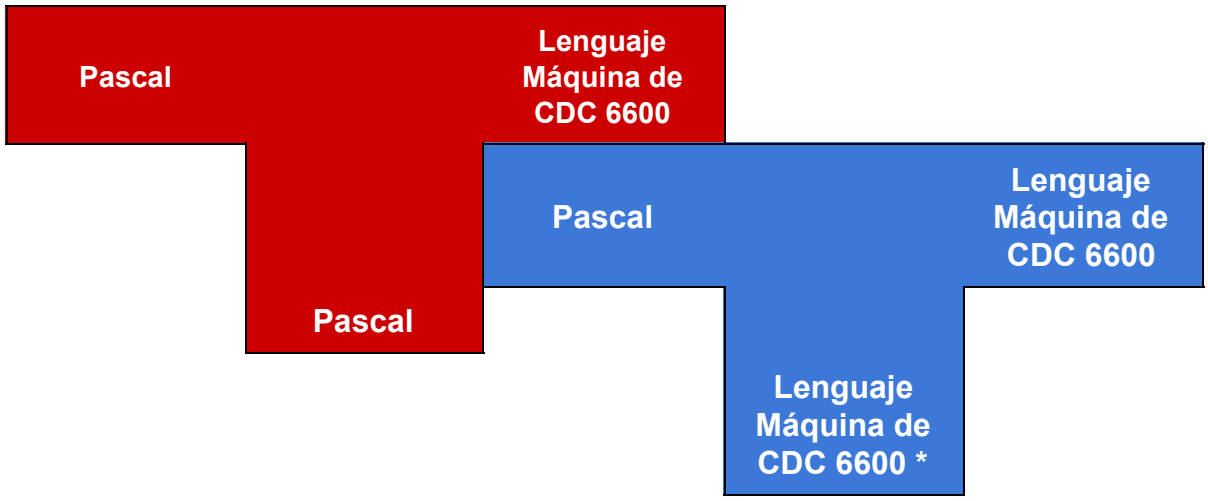
R/ Wirth era un experto en compiladores. Así que decidió ser él mismo quien compilara **a mano** y generar lenguaje máquina de CDC 6600



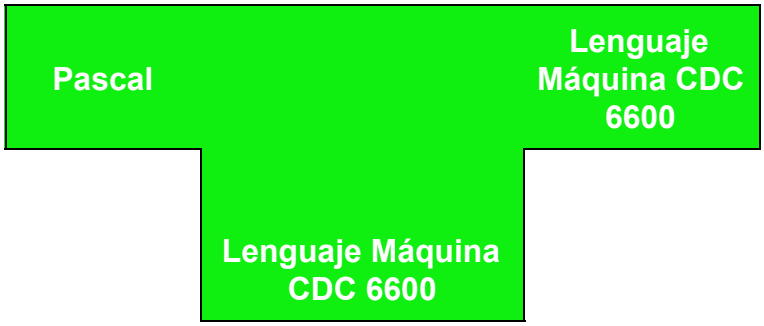
=

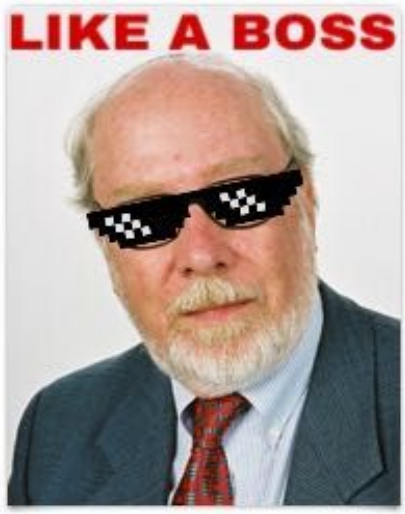


(*)Este Lenguaje máquina resulta ser poco eficiente ya que lo escribió Wirth por sí solo. Para hacerlo eficiente, se recompiló de la siguiente manera:



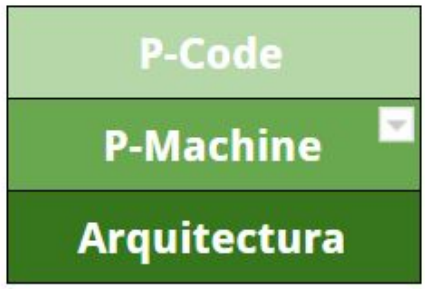
=





Nótese que el host ahora es Lenguaje Máquina CDC 6600 optimizado, y no el poco eficiente que escribió Wirth.

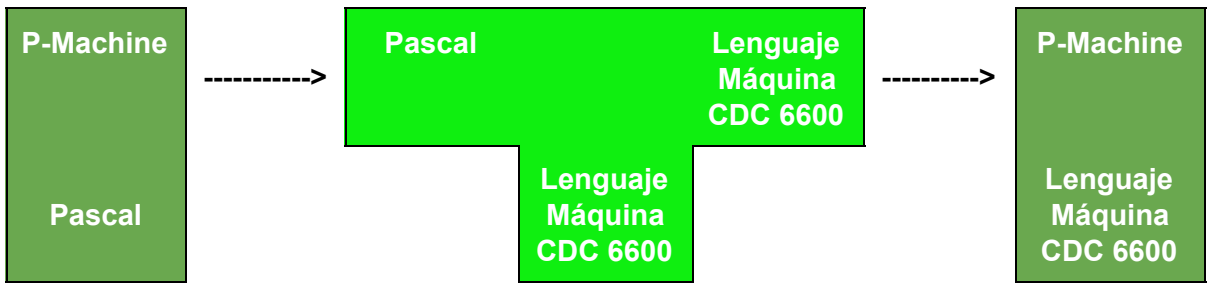
P-Machine y P-Code



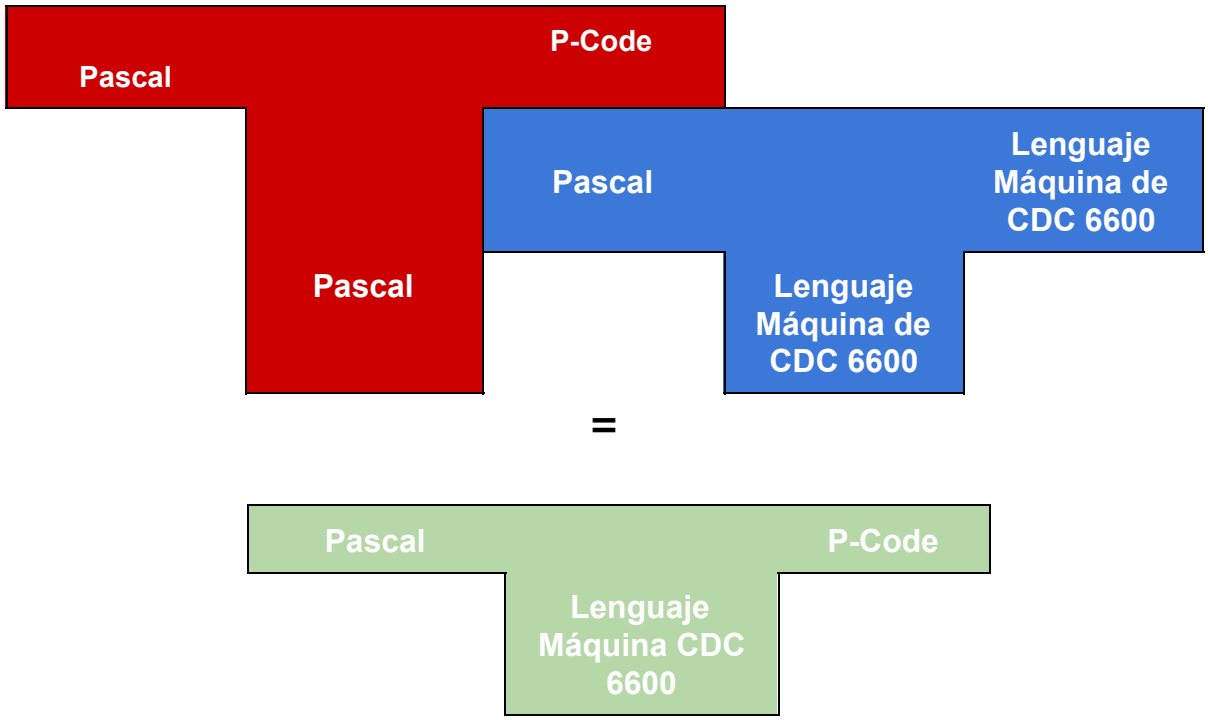
- Portable Machine y Portable Code.
- Es una máquina virtual que corre sobre alguna arquitectura arbitraria.
- El P-Machine era un simulador que ejecutaba P-Code.
- Bastaba con escribir el simulador de la P-Machine
- Hacer una P-Machine era **relativamente fácil.**

Zurich P-Machine

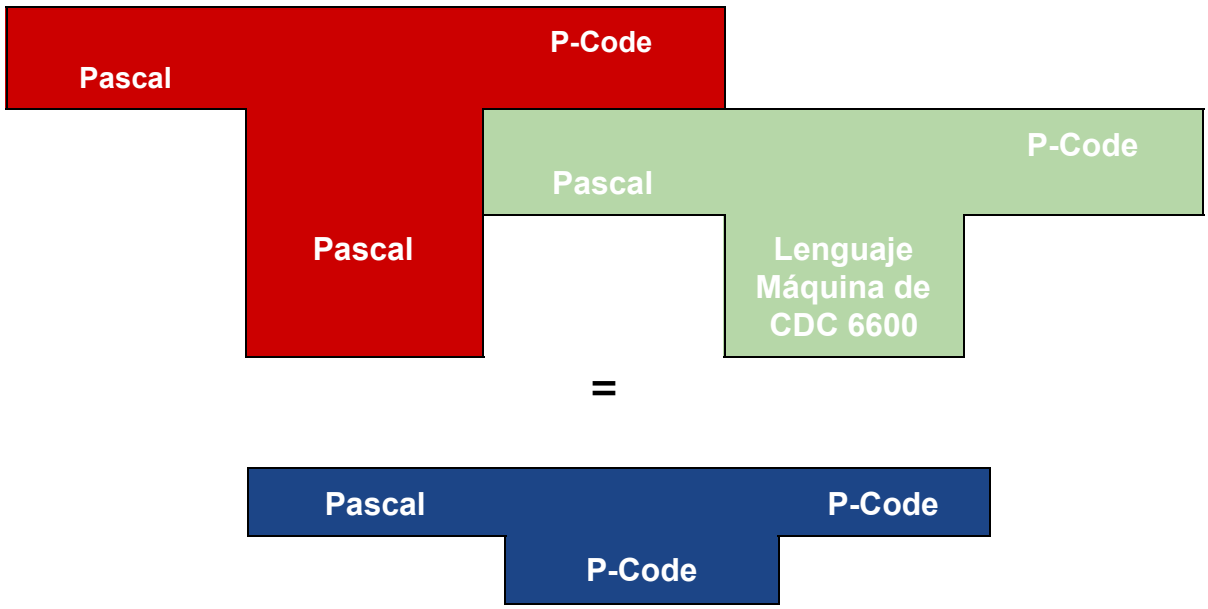
- Debido a que a Wirth le preocupaba la dependencia del compilador con la CDC 6600, ya que Pascal solo funcionaba en esta máquina, su grupo de investigación define una **P-Machine *orientada a pila***.
- Esta se programó en Pascal.



- Para que Pascal generará P-Code, se reescribió el compilador de Pascal:



- Compilamos una última vez para obtener el compilador que corra sobre p-machine



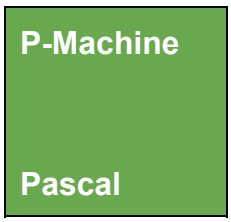
- Como podemos ver, después del largo proceso obtenemos un compilador que recibe Pascal, genere P-Code y está escrito en P-Code

Zurich Compiler Kit

- Wirth y su grupo enviaban su “Compiler Kit” a quien lo pidiera.
- Incluía:

  - ***A:*** *Compilador de Pascal, escrito en Pascal, que genera P-Code*

  - ***B:*** *Compilador de Pascal, escrito en P-Code, que genera P-Code*



  - ***C:*** *P-Machine escrita en Pascal*
- Para que corriera en otras máquinas había que **reescribir en FORTRAN la P-Machine.** No era tan difícil (según el profe).

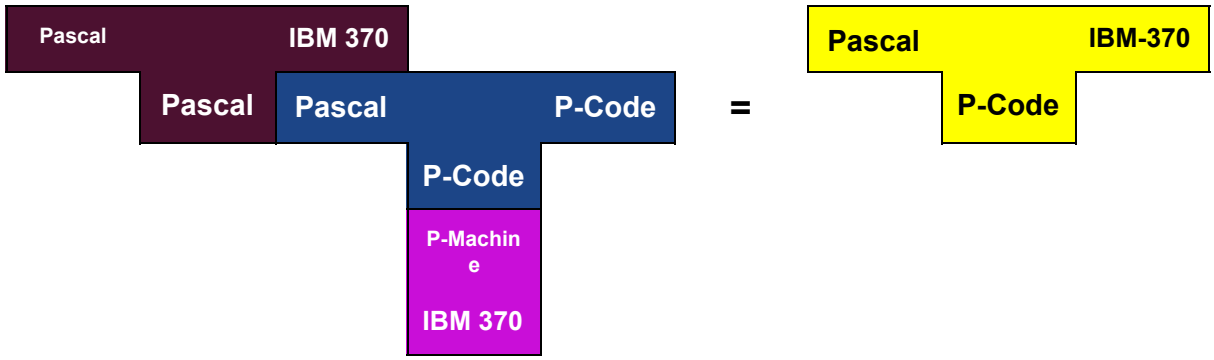
Pasos para reescribir la P-Machine (Aquí se pone bonito).

1. Comenzamos por estudiar:
   1.1. Estudiar al intérprete de la P-Machine escrito en Pascal, este es pequeño y Pascal es un lenguaje muy legible y bonito.
   1.2. Reescribirlo “a mano” en un lenguaje disponible (FORTRAN)

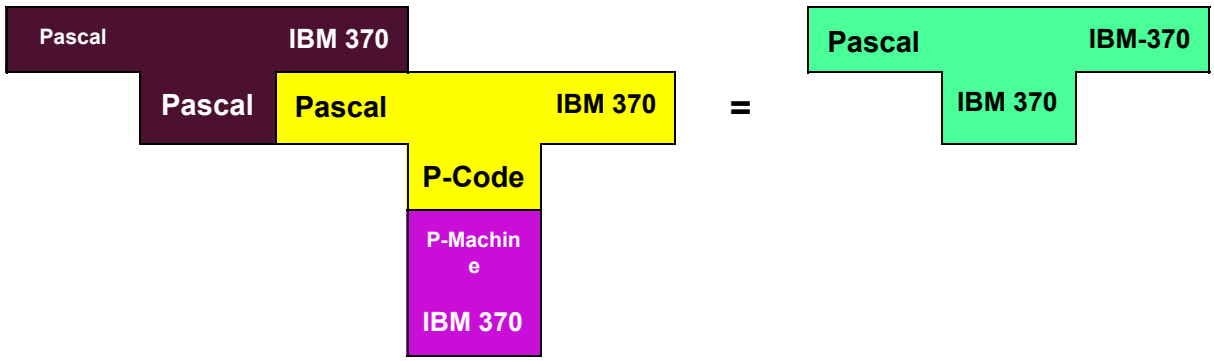
2. Tendremos P-Machine en Lenguaje de IBM 370
   2.1. Podemos utilizar el compilador **B** del Compiler kit para correr en P-Machine

3. Estudiar un poco más:
   3.1. Ahora estudiaremos el compilador **A**, con el objetivo de que en vez de generar P-code, genere lenguaje máquina de la IBM 370. Tendremos el siguiente compilador:

4. Compilamos el nuevo compilador con el compilador **B** del Compiler-Kit:



5. Por último, recompilamos el compilador obtenido (el amarillo) con el obtenido en el paso 3.



Al fin, después de esto, habremos terminado. Cabe mencionar que la máquina IBM 370 fué usada como ejemplo (osea se podía usar otras).
Por último un meme de mi autoría XD