



Profesor: Francisco Torres

Asignatura: Compiladores e Intérpretes

Estudiante: Marlon Agüero Castro

Apuntes de clase

Fecha de apunte: 10 de febrero del 2017

I Semestre 2017

## Lenguaje Maquina

```
*R PC=FFFF SP=FF A=FF B=FF PSW=FF DPTR=FFFF
R0=FF R1=FF R2=FF R3=FF R4=FF R5=FF R6=FF R7=FF
*D 2050,2100
2050 01 02 04 08 10 20 40 80 55 AA 33 01 02 04 08 10 ..... @.U.3.....
2060 20 40 80 55 AA 33 01 02 04 08 10 20 40 80 55 AA @.U.3..... @.U.
2070 33 01 02 04 08 10 20 40 80 55 AA 33 01 02 04 08 3..... @.U.3.....
2080 10 20 40 80 55 AA 33 01 02 04 08 10 20 40 80 55 . @.U.3..... @.U
2090 AA 33 01 02 04 08 10 20 40 80 55 AA 33 01 02 04 .3..... @.U.3.....
20A0 08 10 20 40 80 55 AA 33 01 02 04 08 10 20 40 80 .. @.U.3..... @.
20B0 55 AA 33 01 02 04 08 10 20 40 80 55 AA 33 01 02 U.3..... @.U.3..
20C0 04 08 10 20 40 80 55 AA 33 01 02 04 08 10 20 40 ... @.U.3..... @
20D0 80 55 AA 33 01 02 04 08 10 20 40 80 55 AA 33 01 .U.3..... @.U.3..
20E0 02 04 08 10 20 40 80 55 AA 33 01 02 04 08 10 20 .... @.U.3.....
20F0 40 80 55 AA 33 01 02 04 08 10 20 40 80 55 AA 33 @.U.3..... @.U.3
2100 01 02 04 08 10 20 40 80 55 AA 33 01 02 04 08 10 ..... @.U.3.....
*
```

Este lenguaje es más sencillo que micro programar, ya que, la microprogramación define el lenguaje máquina, por lo tanto, se prefiere usar el lenguaje máquina. A diferencia de micro programar, el lenguaje maquina si fue muy utilizado y se hicieron grandes cosas con este. La responsabilidad del programador con lenguaje máquina, es colocar el programa en RAM, conocer los códigos de operación, modos de direccionamiento, formato de argumentos, calcular tamaños y desplazamientos. Esto seguía sin ser portable.

Con base en la microprogramación, el lenguaje maquina es menos eficiente, pero es mucho más productivo, esto se da porque no todo lo que se hace micro programado se puede hacer en lenguaje máquina.

“La historia de la computación es la historia de la pereza” Francisco Torres.

Esto significa que muchas invenciones se dan porque es aburrido hacer las cosas tal como están, entonces lo mejorar para hacerlo más sencillo y menos aburrido.

## Lenguaje Ensamblador

Se usan nemónicos con el objetivo de no aprenderse la cadena de bit de cada instrucción. En 1950 se inventó el ensamblador. El primer ensamblador se hizo en lenguaje

ejemplo ensamblador: Hola Mundo!

```
DOSSEG
.model small
.stack 100h
.data
msgHello DB "Hola mundo!",13,10,"$"
.code
mov ax,@data          ; Strichpunkt leitet Kommentar ein!
mov ds,ax              ; Datensegment initialisiert

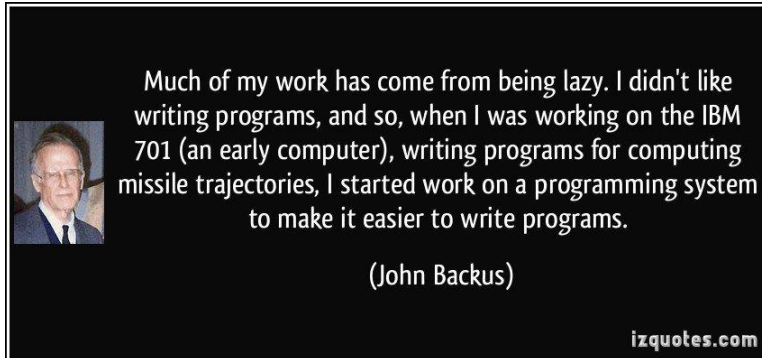
mov dx,offset msgHello ;
mov ah,9               ; Ausgabe eines String
int 21h                ;

mov ax,4C00h           ; Programm beenden
int 21h                ;
END
```

máquina. Este era más sencillo que el lenguaje máquina y mejora ampliamente la productividad, pero mantiene la misma eficiencia de un lenguaje máquina, ya

que cualquier cosa que se puede hacer en lenguaje máquina, se puede hacer en ensamblador. Pero sigue sin ser portable. Las personas de ese tiempo, eran muy buenas trabajando en ensamblador.

### John Backus



Fue un matemático y científico de la computación que vivió de 1924 a 2007, cuando termino su maestría en matemáticas, no sabía dónde conseguir trabajo,

un día caminando, un lugar le llamo la atención entonces decidió entrar y pedir trabajo, aunque ni siquiera sabía que se hacía en ese lugar. Luego de la entrevista se dieron cuenta que era muy inteligente y capaz y se le dio trabajo, aunque él seguía sin saber que hacían en ese lugar. Luego de explicarle que hacían computadoras empezó a trabajar ahí (en IBM) como programador de lenguaje máquina y ensamblador. El primer proyecto que se le asigno fue el de calcular posiciones de la luna en lenguaje maquia. Pero como ya se había mencionado la pereza origina innovaciones, por lo que Backus muy aburrido de lo que hacía, decidió inventar el SpeedCoding, que hacía que programar en lenguaje máquina y en ensamblador fuese más fácil. Pero la pereza lo volvió a atacar y quería hacer todo aún más fácil, por lo que en 1953 propone FORTRAN que significa Formula Translate, quería hacerlo para la IBM 704. Pero en ese tiempo se pensaba en muchos retos que podía afrontar este proyecto, como si es posible hacerlo o si el código es tan bueno como el de los mejores programadores de ensamblador.

En 1956 termino el proyecto y creo FORTRAN, duro tanto ya que fue el primero en hacerlo, en crear un lenguaje de alto nivel, no había donde consultar sobre esto y además, lo programo muy posiblemente en ensamblador y lenguaje máquina. FORTRAN en la actualidad sigue existiendo y es utilizado, aunque ha recibido muchas actualizaciones. En 1958 creo BNF que sirve para describir cualquier lenguaje de programación libre de contexto. Significa Backus Normal Form, aunque recibió aportes de Peter Naur años después, y se le cambio el

nombre a Backus Naur Form. En 1977 crea FP que es el primer lenguaje de programación funcional. Y gano un Turing Award en 1977.

### John McCarthy

Matemático y científico de la computación, vivió de 1927 a 2011. Fue profesor en MIT, Stanford y Princeton. Fue influenciado por Von Neumann y es un pionero



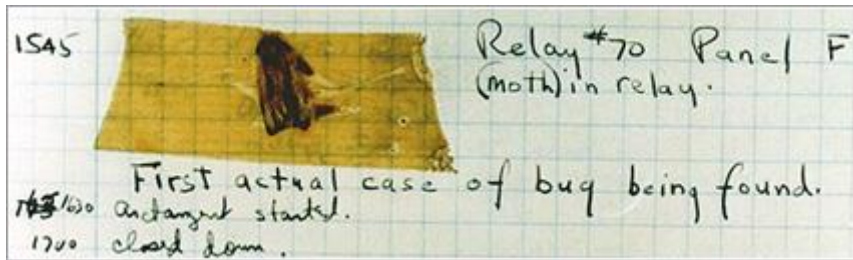
de la inteligencia artificial. Invento LIPS y conceptos como el garbage collection. Fue parte del equipo que creo Algol y gano un Turing Award en 1971.

### Grace Murray Hopper



Matemática y científica de la computación de 1906 a 1992, doctorado en Yale, fue almirante de la Marina. Muchos opinan que pudo ser la primera programadora, de lo que se conoce hoy en día. Ella creo Cobol en 1959, que muchos piensan que era muy malo, pero fue demasiado popular. Ella antes de Cobol creo algo que confunden con un lenguaje de alto nivel, pero era como speedcoding que se llamaba Flow Matic. Ella también por error creo la palabra compilador, ya que pensaba que

cuando el programa traducía, iba a traer pedazo por pedazo cada significado, luego se dio cuenta del error, pero el termino quedo. Ella también creo la palabra



debugging para encontrar errores, pero esto se originó porque literalmente una polilla se encontró dentro de la Mark II en 1945 y se documentó. Pero como dato curioso antes de que pasara esto, el termino bug ya se usaba y procedía de la telegrafía, pero este caso es famoso porque fue la primera vez que se documentó un error por un bug real. No tuvo un Turing Award, posiblemente porque Cobol no fue muy aceptado por la academia y también por el machismo de la época.

### Peter Naur

Astrónomo y científico de la computación. Nació en Dinamarca en 1928 y murió el 3 de enero del 2016. Tuvo un doctorado en astrofísica en 1957, como ya se había mencionado tuvo muchos aportes al BNF, fue el líder del Comité que creo ALGOL 60. Dio muchas contribuciones a los lenguajes de programación actuales. Y gano un Turing Award en el 2005.

### Los 4 grandes

#### FORTRAN (Formula Translate)

Primer lenguaje de alto nivel de uso generalizado, era portable, hecho por John Backus, en IBM 1957. Se utilizaban tarjetas perforadas, aun hoy se utiliza bastante, pero sin las tarjetas perforadas. Es el favorito para HPC (High Performance Computing). Cada tarjeta perforada era una línea de código. Ha evolucionado en muchas versiones desde su creación. Las tarjetas perforadas utilizaban código Hollerith.

```

1      PROGRAM PRINCIPAL
2      PARAMETER (TAMMAX=99)
3      REAL A(TAMMAX)
4      10  READ (5,100,END=999) K
5      100 FORMAT(I5)
6          IF (K.LE.0.OR K.GT.TAMMAX) STOP
7          READ *,(A(I),I=1,K)
8          PRINT *,(A(I),I=1,K)
9          PRINT * , 'SUMA=', SUM(A,K)
10         GO TO 10
11      99  PRINT * , "Todo listo"
12         STOP
13         END
14  SUBPROGRAMA DE SUMATORIA EN C
15  FUNCTION SUM(V,N)
16      REAL :: V(N) ! Declaración de estilo nuevo
17      SUM = 0.0
18      DO 20 I = 1,N
19          SUM = SUM + V(I)
20      20  CONTINUE
21      RETURN
22      END

```



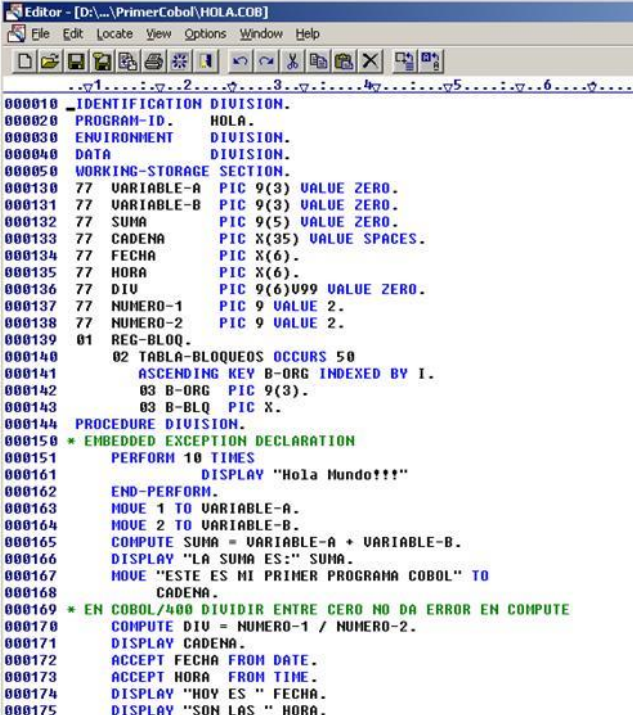
## LISP (LISt Processing)

Fue desarrollado por McCarthy en 1958, es prefija, interpretado, utilizaba muchos paréntesis, era orientado a inteligencia artificial, paradigma funcional y todo eran listas, derivados de LISP nacen Scheme, CommonLISP y Clojure.

```
1 ;lisp
2 >; Guardar valores como una lista de caracteres
3 >(define (SumarSiguiente V)
4   (cond ((null V) (progn (print "Suma=") 0))
5         (T (+ ( SumarSiguiente (cdr V)) (car V) ) ))
6 SUMARSIGUIENTE
7 >; Crear vector de valores de entrada
8 (defun ObtenerEntrada(f c)
9   (cond ((eq c 0) nil)
10         (T (cons (read f) (ObtenerEntrada f (- c 1))))))
11 OBTENERENTRADA
12 >(defun Hazlo()
13   (progn
14     ( setq archivoent (open "lisp.data"))
15     ( setq arreglo (ObtenerEntrada archivoent (read archivoent)))
16     (print arreglo)
17     (print (SumarSiguiente arreglo))))
18 HAZLO
19 >Hazlo
20
21 (1 2 3 4)
22 "Suma="
23 10
24 10
```

## COBOL (Common Business-Oriented Language)

Creado por Hopper en 1959, pretendía parecerse mucho al inglés, este debería ser sencillo para todos y autodocumentado, pero resultó ser totalmente lo contrario y muy grande. Tenía Muchas palabras reservadas, por todo esto, fue muy criticado por la academia. Pero hoy día sigue vivo, comercialmente fue un éxito, muy posiblemente porque se comieron la hablada de Hopper. Todo el código de COBOL de hoy en día nadie lo



```
Editor - [D:\...\PrimerCobol\HOLA.COB]
File Edit Locate View Options Window Help

000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. HOLA.
000030 ENVIRONMENT DIVISION.
000040 DATA DIVISION.
000050 WORKING-STORAGE SECTION.
000130 77 VARIABLE-A PIC 9(3) VALUE ZERO.
000131 77 VARIABLE-B PIC 9(3) VALUE ZERO.
000132 77 SUMA PIC 9(5) VALUE ZERO.
000133 77 CADENA PIC X(35) VALUE SPACES.
000134 77 FECHA PIC X(6).
000135 77 HORA PIC X(6).
000136 77 DIV PIC 9(6)V99 VALUE ZERO.
000137 77 NUMERO-1 PIC 9 VALUE 2.
000138 77 NUMERO-2 PIC 9 VALUE 2.
000139 01 REG-BLOQ.
000140 02 TABLA-BLOQUEOS OCCURS 50
000141 ASCENDING KEY B-ORG INDEXED BY I.
000142 03 B-ORG PIC 9(3).
000143 03 B-BLQ PIC X.
000144 PROCEDURE DIVISION.
000150 * EMBEDDED EXCEPTION DECLARATION
000151 PERFORM 10 TIMES
000161 DISPLAY "Hola Mundo!!!"
000162 END-PERFORM.
000163 MOVE 1 TO VARIABLE-A.
000164 MOVE 2 TO VARIABLE-B.
000165 COMPUTE SUMA = VARIABLE-A + VARIABLE-B.
000166 DISPLAY "LA SUMA ES:" SUMA.
000167 MOVE "ESTE ES MI PRIMER PROGRAMA COBOL" TO
000168 CADENA.
000169 * EN COBOL/400 DIVIDIR ENTRE CERO NO DA ERROR EN COMPUTE
000170 COMPUTE DIV = NUMERO-1 / NUMERO-2.
000171 DISPLAY CADENA.
000172 ACCEPT FECHA FROM DATE.
000173 ACCEPT HORA FROM TIME.
000174 DISPLAY "HOY ES " FECHA.
000175 DISPLAY "SON LAS " HORA.
```

quiere tocar. Par el 2016 el 80% del código este hecho en COBOL. Tenía solo variables globales, se dividía en párrafos en vez de procedimientos.

## ALGOL (Algorithmic Language)

Hecho en 1958 por el comité dirigido por Peter Neur, tenía el propósito de resolver defectos de FORTRAN, primero en usar código con una estructura de begin-end, utilizaba llamadas por valor y referencia, también tenía variables locales e

introduce then/else, while y for. Este a pesar de introducir muchos conceptos aun usados, no tuvo mucho éxito comercial, pero si es muy usado por académicos. Luego de su creación inicio los Algol-like language que son lenguajes de programación basados en los conceptos de ALGOL, como pascal, C, C++, etc.

### Lenguajes de Alto Nivel

Existen desde 1956 con la creación de FORTRAN. Son fáciles de programar, por lo que tiene mucha productividad, pero poca eficiencia, ya que no se puede hacer todo lo que sí se puede hacer en ensamblador.