UNIVERSIDAD
DE LIMA

Lenguajes de Programación

Historia y Criterios de Diseño

Hernan Quintana (hquintan@ulima.edu.pe)

Normas

Asistencia a clases

- Se tomará lista a 20 minutos de iniciada la clase y también al final.
- No estar en alguna lista invalida su asistencia.
- No se puede justificar inasistencias a no ser por lo siguiente:
  - Faltar más de 1 semana (con secretaría académica)
  - Eventos deportivos acreditados por secretaría académica.
  - Reunión de delegados (de darse el caso)

Comunicación

- Dudas, reclamos y observaciones (fuera de clase o asesoría): Por **correo institucional** al profesor o al JP.

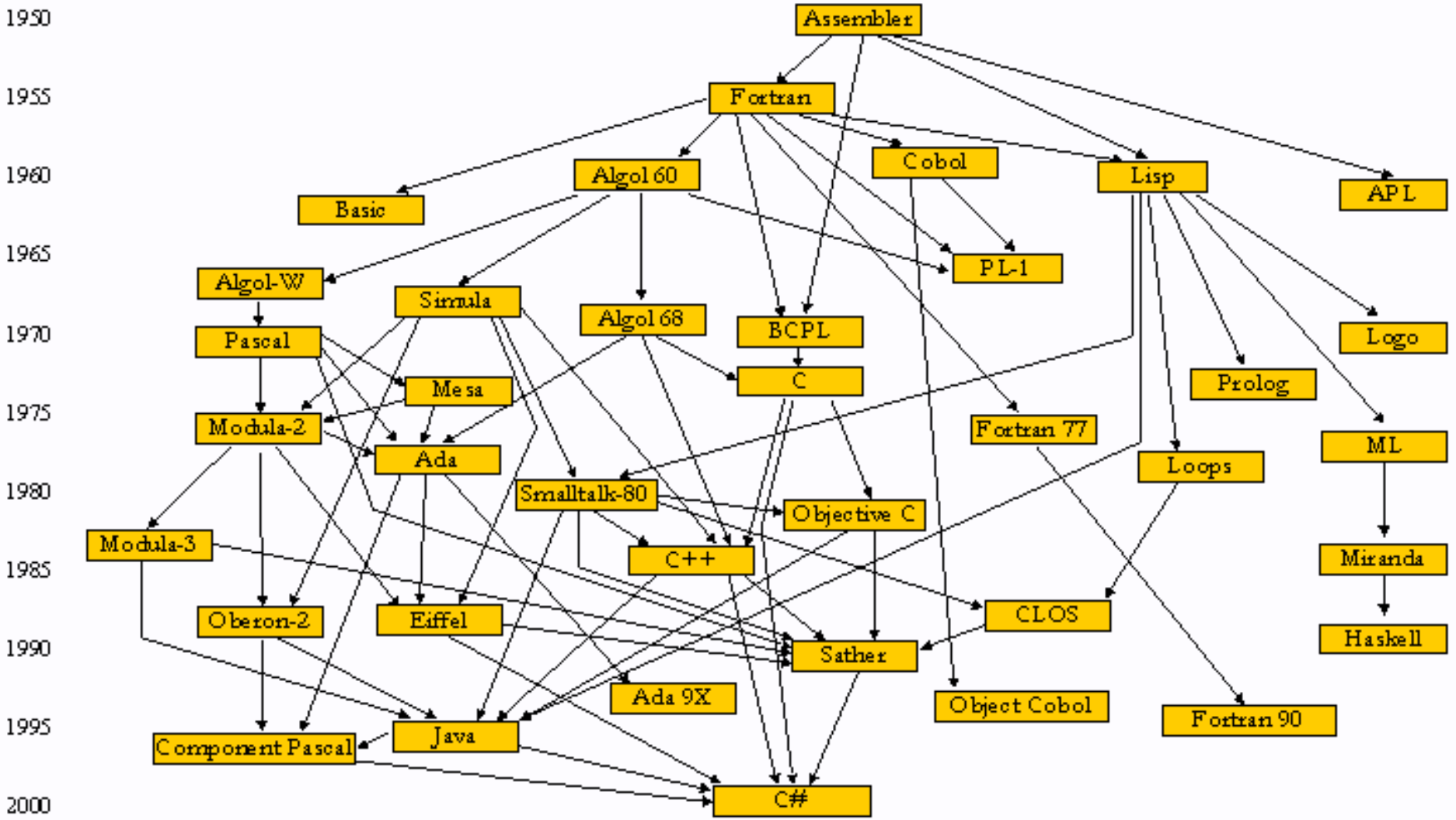
Código de Ética

- La Universidad de Lima fomenta la innovación, el desarrollo científico y el pensamiento crítico de su comunidad. Asimismo, promueve la protección y el respeto de las obras que son producto del esfuerzo académico en todas sus manifestaciones. En tal sentido, se considera como plagio cualquier acción que utilice el trabajo de otra persona o el resultado del empleo de inteligencia artificial como propio, sin permiso o sin mencionar la fuente. Esta acción constituye una transgresión a los derechos de autor y a las normas éticas que sustentan el trabajo académico. Asimismo, se considera plagio en las evaluaciones académicas utilizar o acceder a medios o información distintos a los permitidos para el desarrollo de la evaluación.

Estructura de la Evaluación

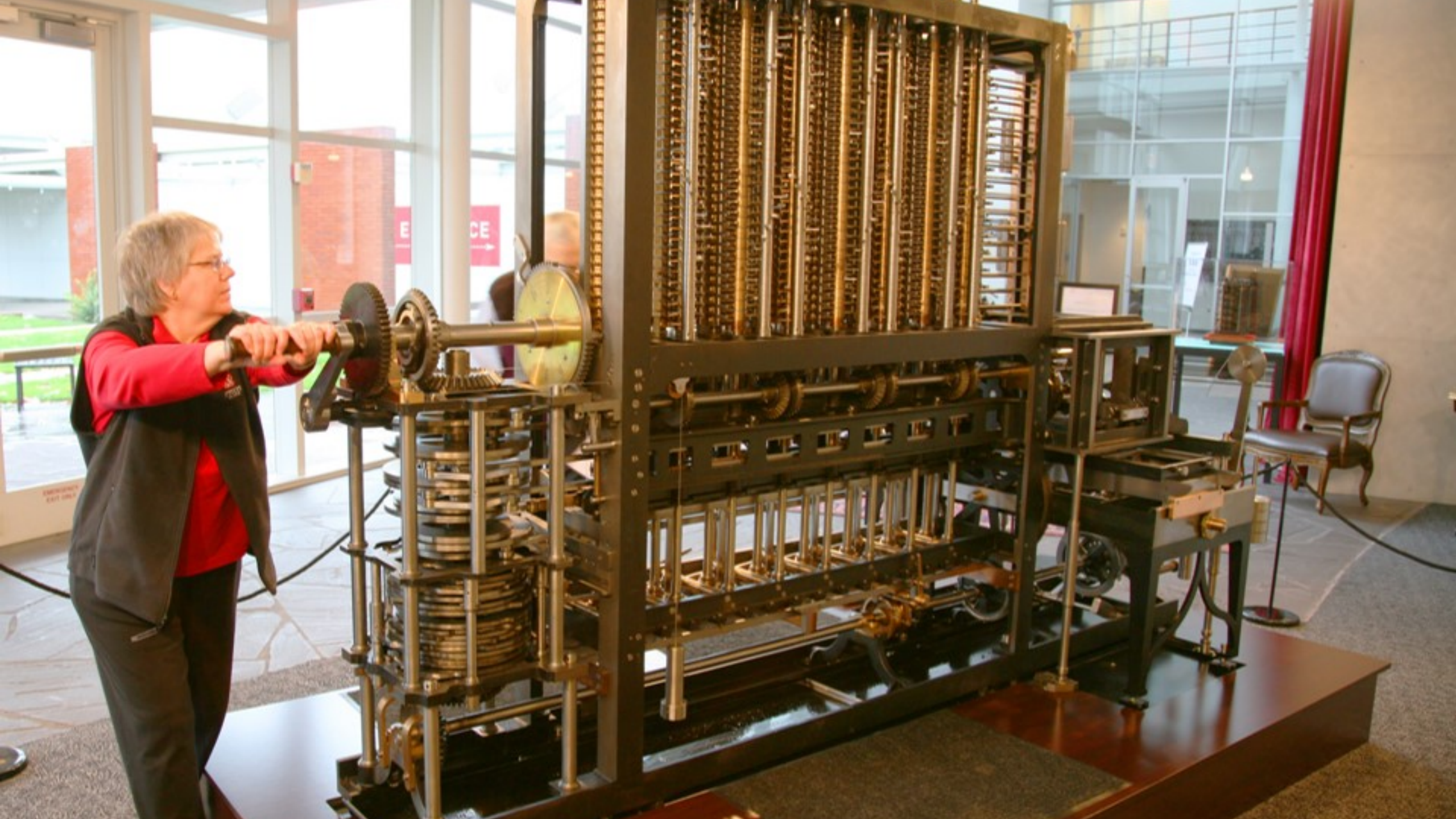
Historia

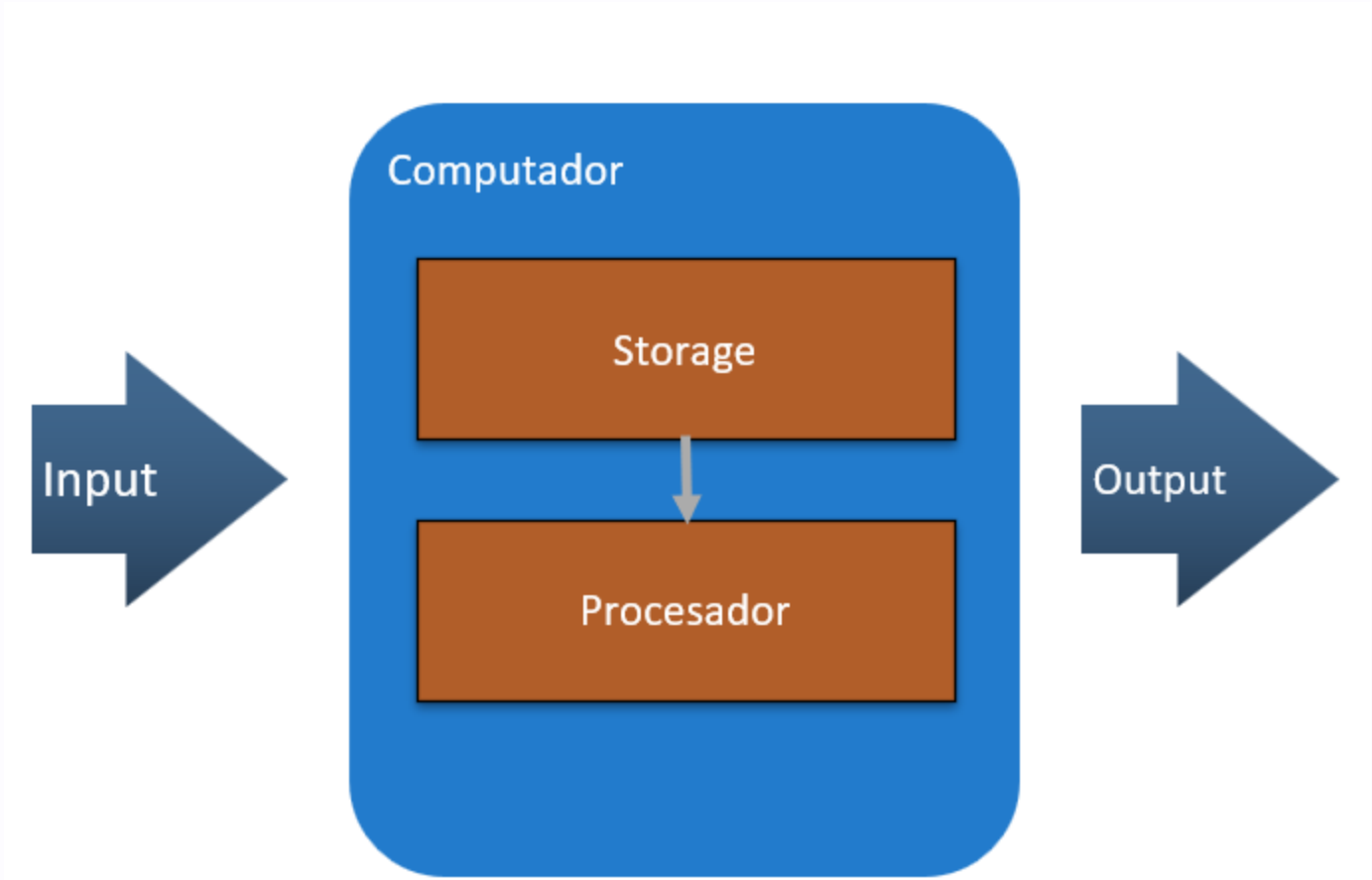
Programming Language Family Tree



Pre-computación

- En la historia hubieron muchos intentos de crear una máquina que pudiera realizar cálculos complejos.
- Ada Lovelace, define teóricamente el primer lenguaje de programación.
- Charles Babbage, que crea la una máquina mecánica que permitía calcular derivadas, define el concepto de máquina de propósito general.





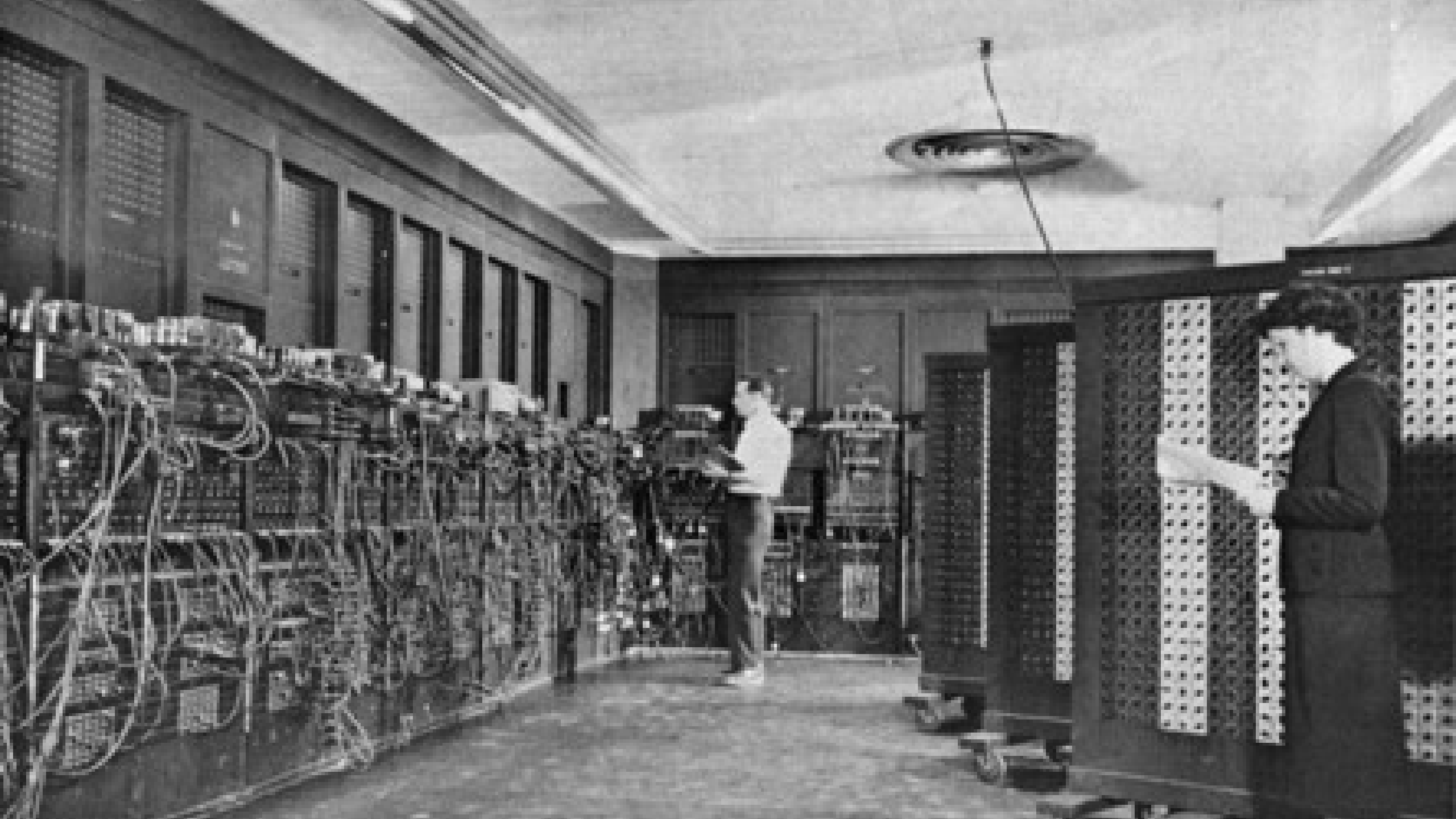
Arquitectura de Von Neumann

Nace el computador

1. Recibe data (input)
2. Almacena la data recibida (storage)
3. Procesa la data
4. Devuelve los resultados (output)

Primer computador

- Dada la segunda guerra mundial, estas máquinas se perfeccionan y nace el concepto de computadoras que eran máquinas de propósito general.
- El problema es que programarlas para realizar alguna tarea, consistía en ajustar distintos cables y elementos de hardware (labor muy compleja).



Lenguaje máquina

- Ya no sería necesario manipular cables, sino se ingresarían un conjunto de valores binarios a la memoria del computador (storage).
- Estos valores binarios serán códigos especiales que el computador entenderá para realizar alguna tarea.

opcode	Instrucciones
0010	001000000100
0010	010000000100
0001	011001000010
0011	011000000011
1111	000000100101
0000	000000000101
0000	000000000110
0000	000000000000

0010 001 000000100

0010 (opcode)	Copiar un valor (número) de una dirección de memoria a un registro.
Primeros 3 bits 001 (1)	Número del registro destino
Siguientes 9 bits 000000100 (4)	Offset a partir de la siguiente instrucción donde se encuentra almacenado el valor (origen).

opcode Instrucciones

0010001000000100

0010010000000100

0001011001000010

0011011000000011

1111000000100101

0000000000000101

0000000000000110

0000000000000000

0010 010 000000100

0010 (opcode)	Copiar un numero de una dirección de memoria a un registro.
Primeros 3 bits 010 (2)	Número del registro destino.
Siguientes 9 bits 000000100 (4)	Offset a partir de la siguiente instrucción donde se encuentra almacenado el valor (origen).

opcode Instrucciones

0010001000000100

0010010000000100

0001011001000010

0011011000000011

1111000000100101

0000000000000101

0000000000000110

0000000000000000

0001 011 001 0 00 010

0001 (opcode)	Sumar
Primeros 3 bits 011 (3)	Número del registro destino
Siguientes 3 bits 001 (1)	Número del registro operando 1
Siguiente 1 bit 0	Bit que nos indica tipo de suma
2 bits 00	Bits que se dejan en 0
Últimos 3 bits 010 (2)	Número del registro operando 2

opcode Instrucciones

0010001000000100

0010010000000100

0001011001000010

0011011000000011

1111000000100101

0000000000000101

0000000000000110

0000000000000000

0011 011 000000011

0011 (opcode)	Mover de un registro a la memoria
Primeros 3 bits 011 (3)	Número del registro origen
Últimos 9 bits 000000011 (3)	Offset a la dirección donde se almacenará (destino).

1111 000000100101

0010001000000100

0010010000000100

0001011001000010

0011011000000011

1111000000100101

0000000000000101

0000000000000110

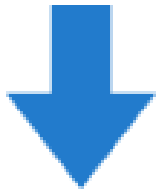
0000000000000000

1111 (opcode)	Halt (fin de programa)

Lenguaje ensamblador

- Facilita la programación del lenguaje máquina.
- Necesita de un ensamblador (assembler) que es un herramienta de software.
- El ensamblador traduce este nuevo lenguaje simbólico (llamado lenguaje ensamblador) a lenguaje de máquina binario.

0010 001 000000100



LD R1, FIRST

0010 (opcode)	Copiar un valor (número) de una dirección de memoria a un registro.
Primeros 3 bits 001 (1)	Número del registro destino
Siguientes 9 bits 000000100 (4)	Offset a partir de la siguiente instrucción donde se encuentra almacenado el valor (origen).

Lenguaje máquina

```
0010001000000100
0010010000000100
0001011001000010
0011011000000011
1111000000100101
0000000000000101
0000000000000110
0000000000000000
```

=

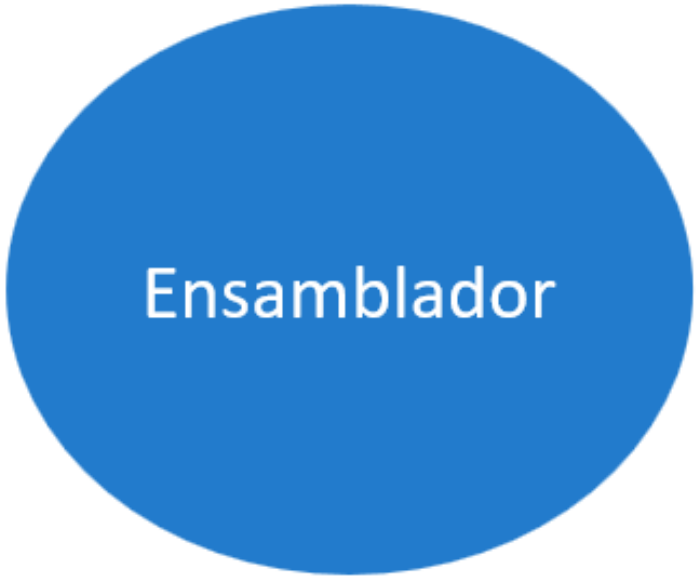
Lenguaje ensamblador

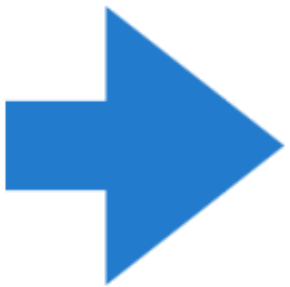
```
.ORIG x3000
LD      R1, FIRST
LD      R2, SECOND
ADD     R3, R2, R1
ST      R3, SUM
HALT
FIRST           .FILL    #5
SECOND          .FILL    #6
SUM             .BLKW    #1
```

Assembler

Lenguaje Ensamblador

```
.ORIG x3000
LD        R1, FIRST
LD        R2, SECOND
ADD       R3, R2, R1
ST        R3, SUM
HALT
FIRST     .FILL     #5
SECOND    .FILL     #6
SUM       .BLKW     #1
```





Lenguaje Máquina

```
0010001000000100
0010010000000100
0001011001000010
0011011000000011
1111000000100101
0000000000000101
0000000000000110
0000000000000000
```

Traduce

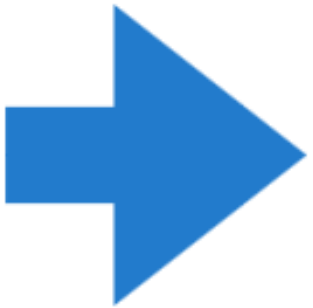
Notacion Algebraica

- El lenguaje ensamblador tenía también complejidad ya que Se tenía que conocer cómo funcionaba el procesador, cómo trabajar con operaciones binarias, etc.
- Como respuesta a esto se crearon los conocidos como lenguajes de alto nivel que tomaron la notación matemática (algebraica) para definir sus operaciones.
- El primer lenguaje de alto nivel se llamó FORTRAN (FORmula TRANslation), desarrollado por John Backus en IBM.

Fortran

Lenguaje FORTRAN

```
int first = 5;
int second = 6;
int sum = first + second;
```

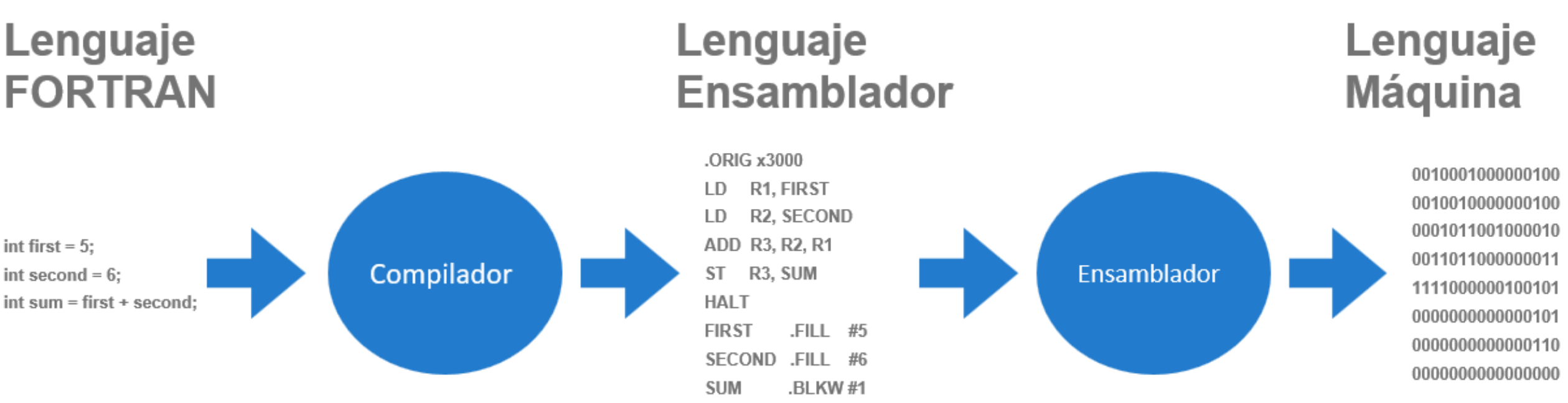


Assembler

```
.ORIG x3000
LD      R1, FIRST
LD      R2, SECOND
ADD     R3, R2, R1
ST      R3, SUM
HALT
FIRST   .FILL   #5
SECOND  .FILL   #6
SUM     .BLKW   #1
```

Compilador

Herramienta (software) que permite generar (no traducir) lenguaje máquina a partir de lenguaje FORTRAN a lenguaje ensamblador.


Lenguaje FORTRAN
int first = 5;
int second = 6;
int sum = first + second;
Compilador
Lenguaje Ensamblador
.ORIG x3000
LD R1, FIRST
LD R2, SECOND
ADD R3, R2, R1
ST R3, SUM
HALT
FIRST .FILL #5
SECOND .FILL #6
SUM .BLKW #1
Ensamblador
Lenguaje Máquina
0010001000000100
0010010000000100
0001011001000010
0011011000000011
1111000000100101
0000000000000101
0000000000000110
0000000000000000

Lenguajes Estructurados

- Nacen a partir de la necesidad de ejecutar algoritmos más complejos que operaciones matemáticas.
- Introducen estructuras complejas (para datos y para control).
  - Para datos: arreglos.
  - Para control: loops, condicionales.
- El primer lenguaje fue el ALGOL (ALGOrithmic Language) que comenzó a utilizarse en los 60s.

Paradigmas

Qué son?

- Los lenguajes de programación pueden agruparse en paradigmas.
- Los paradigmas generalizan algunas propiedades de los lenguajes que permiten su agrupación.
- El uso de estos paradigmas se verá influenciado por el contexto del problema donde se requiera aplicar el lenguaje.
- Son 4 paradigmas: Imperativo, Funcional, Lógico y Orientado a Objectos.

Paradigma Imperativo

- Sigue a cabalidad el modelo de Von Neumann de ejecución secuencial de instrucciones.
- Su característica principal es que se basa en la ejecución de sentencias que representas comandos que la computadora ejecutará. A estos comandos se le conoce como imperatives en inglés.
- Ejemplos: C, Pascal, Cobol, FORTRAN, Python?

Paradigma Funcional

- Se base en el concepto matemático de las funciones (una función devuelve un resultado dado ciertos parámetros de entrada).

```
definir( resultado, 0 )
definir( resultado, sumar( 4, 4) , resultado )
```

  - No hay sentencias, solamente existen funciones.
  - Ejemplos: Scheme, Haskell, Scala

Paradigma Lógico

- Paradigma basado en operaciones de lógica simbólica.
- Ejm: Prolog

```
length([], 0).
length([A|B],N) :- length(B,M), N is M+1 .

smallest([A],A) :- !.
smallest([A|B],N) :- smallest(B,N), N<A, !.
smallest([A| _ ], A).
```

Paradigma Orientado a Objetos

- Escribir código reusable mediante la encapsulación y el polimorfismo.
- Es una extensión del paradigma imperativo que permite la organización del código en partes pequeñas (componentes) poniendo énfasis a las interacciones entre estos así como en la matenibilidad del código.
- Ejemplos: C++, Java, Smalltalk, Ruby, Python, etc.