UNIVERSIDAD DE Belgrano

BUENOS AIRES - ARGENTINA

FUNDAMENTOS DE INFORMATICA

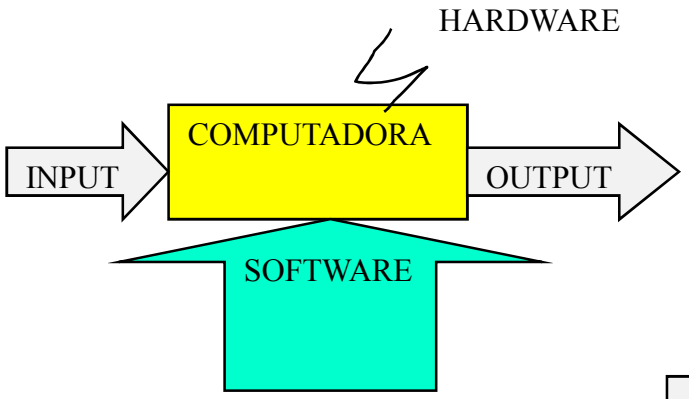
MODULO IV:

Software

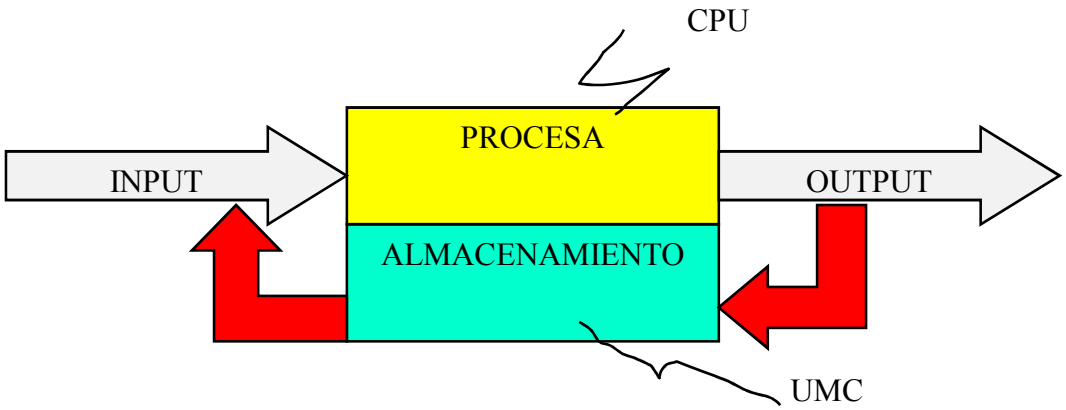
ALGORITMOS Y PROGRAMAS

TODA RESOLUCIÓN DE UN PROBLEMA CONSTA DE LOS SIGUIENTES PASOS:

A. DEFINICIÓN & ANALISIS DEL PROBLEMA
B. DISEÑO DEL ALGORITMO.
C. TRANSFORMAR EL ALGORITMO EN PROGRAMA CODIFICADO.
D. EJECUCI´NO Y VALIDACIÓN DEL PROGRAMA.



PROCESO DE LA INFORMACION



ALGORITMOS Y PROGRAMAS

TODO MÉTODO DE PROGRAMACION SURGE DE UNA INFERENCIA LOGICA

TODO ALGORITMO ES INDEPENDIENTE DEL LENGUAJE, PERO TIENE UNA DEPENDENCIA DEL PARADIGMA DE PROGRAMACION

CARACTERISTICAS DE TODO ALGORITMO

A. PRECISO.
B. DEFINIDO => UN RESULTADO.
C. FINITO

TIPOS DE ALGORITMOS

A. PSEUDOCÓDIGO.
B. LÓGICO.
C. BLOQUE – TEOREMA DE CHAPPIN.

ALGORITMOS EJEMPLOS

Ejercicio 1: Algoritmo para la obtención de un crédito.

Ejercicio 2: Desarrollar un Algoritmo sobre como se prueba Si un número es Primo.

Ejercicio 3. Sumatoria de los números pares de [0, 1000]

SOFTWARE

Suma total de los programas de cómputos, procedimientos, Reglas y documentación; más los datos asociados que forman Parte de las operaciones de un Sistema de Computo (IEEE standard 729)

Características:

- Elemento lógico.
- Difícil de Medir
- Se desarrolla, no se fabrica
- No se rompe, se deteriora.
- Debe actualizarse y mantenerse.
- Normalmente cerrado, baja reutilización

Atributos:

- Mantenibilidad
- Confiabilidad.
- Eficiencia.
- Usabilidad.

INGENIERIA DE SOFTWARE

Conjunto de Técnicas y Herramientas que controlan el proceso total De desarrollo de software, y suministra las bases de calidad para la construcción de Software en tiempo y con eficiencia.

Elementos:

- **Métodos y Técnicas.** ***Planificación y Estimación, Análisis de Requisitos, Diseño de Programas, Estructuras de Datos, Documentación, Pruebas-Testing, Mantenimiento.***
- **Herramientas. Upper Case (Análisis de Requerimientos y diseño) Lower Case (Programación, depuración y pruebas)**
- **Procedimientos. Secuencia de aplicación de métodos y herramientas, etc.**
- **Paradigmas. Filosofía de Construcción de Software.**
  - **a. Desarrollo en cascada.**
  - **b. Desarrollo en Espiral.**
  - **c. Desarrollo por prototipos.**
  - **d. Desarrollo incremental.**
  - **e. Desarrollo en V.**
  - **f. Desarrollo Orientado a Objetos.**

INGENIERIA DE SOFTWARE

PARADIGMA 1: DESARROLLO EN CASCADA:

- **Ordenación rigurosa de las etapas del ciclo de vida del software**
- **Cada etapa sólo se inicia al finalizar la anterior**

Etapas:

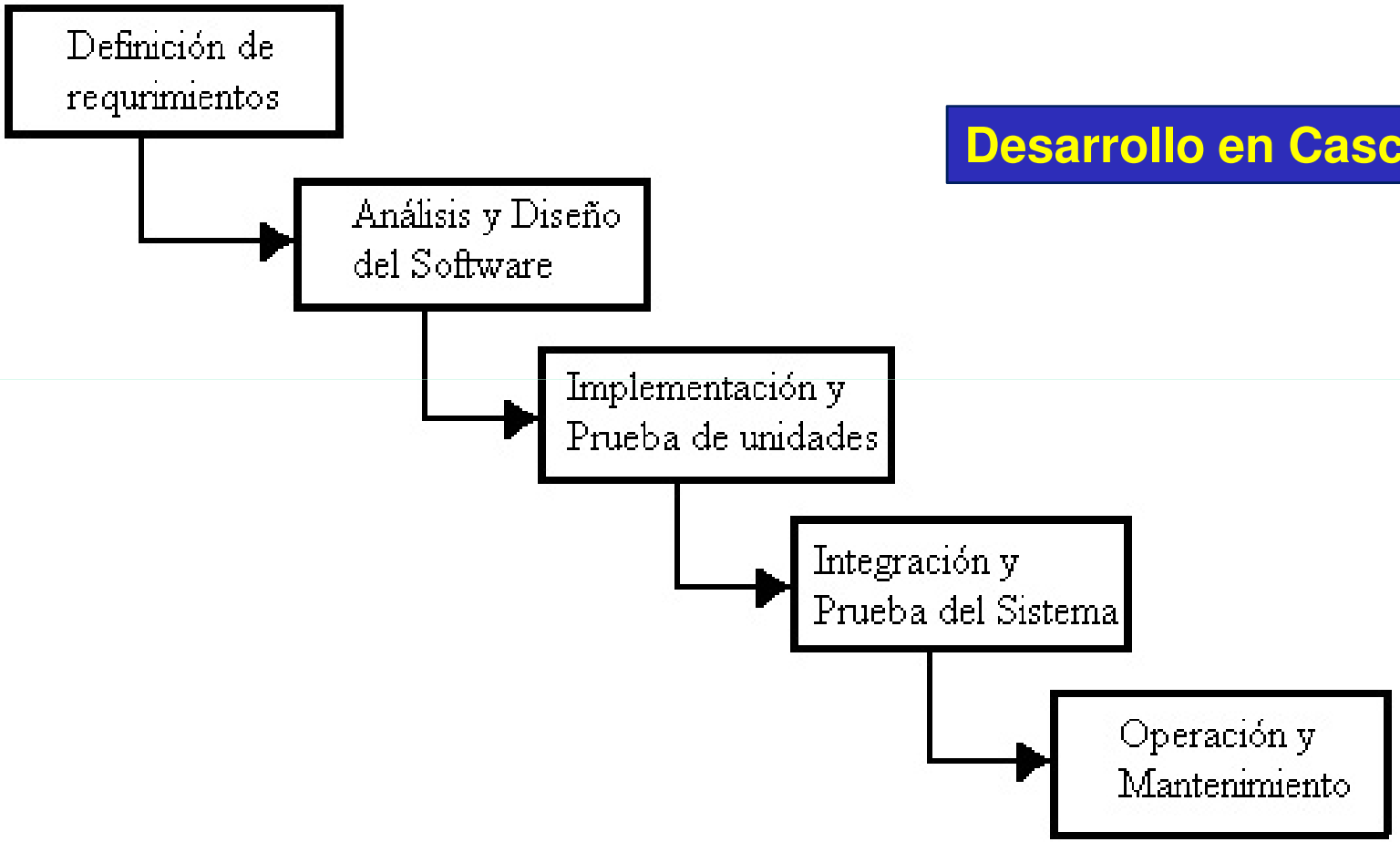
- **Análisis de Requisitos.**
- **Diseño.**
- **Programación – Codificación.**
- **Prueba – Testing.**
- **Implantación.**
- **Mantenimiento.**

Comentarios:

- **Más criticado, pero el más usado.**
- **Esta basada en convertir al Desarrollo de software en una cadena de producción.**
- **Cuando se encuentran errores, durante el testing. Se vuelve a los pasos anteriores. Esto aumenta costos de todo tipo.**

INGENIERIA DE SOFTWARE

Desarrollo en Cascada



INGENIERIA DE SOFTWARE

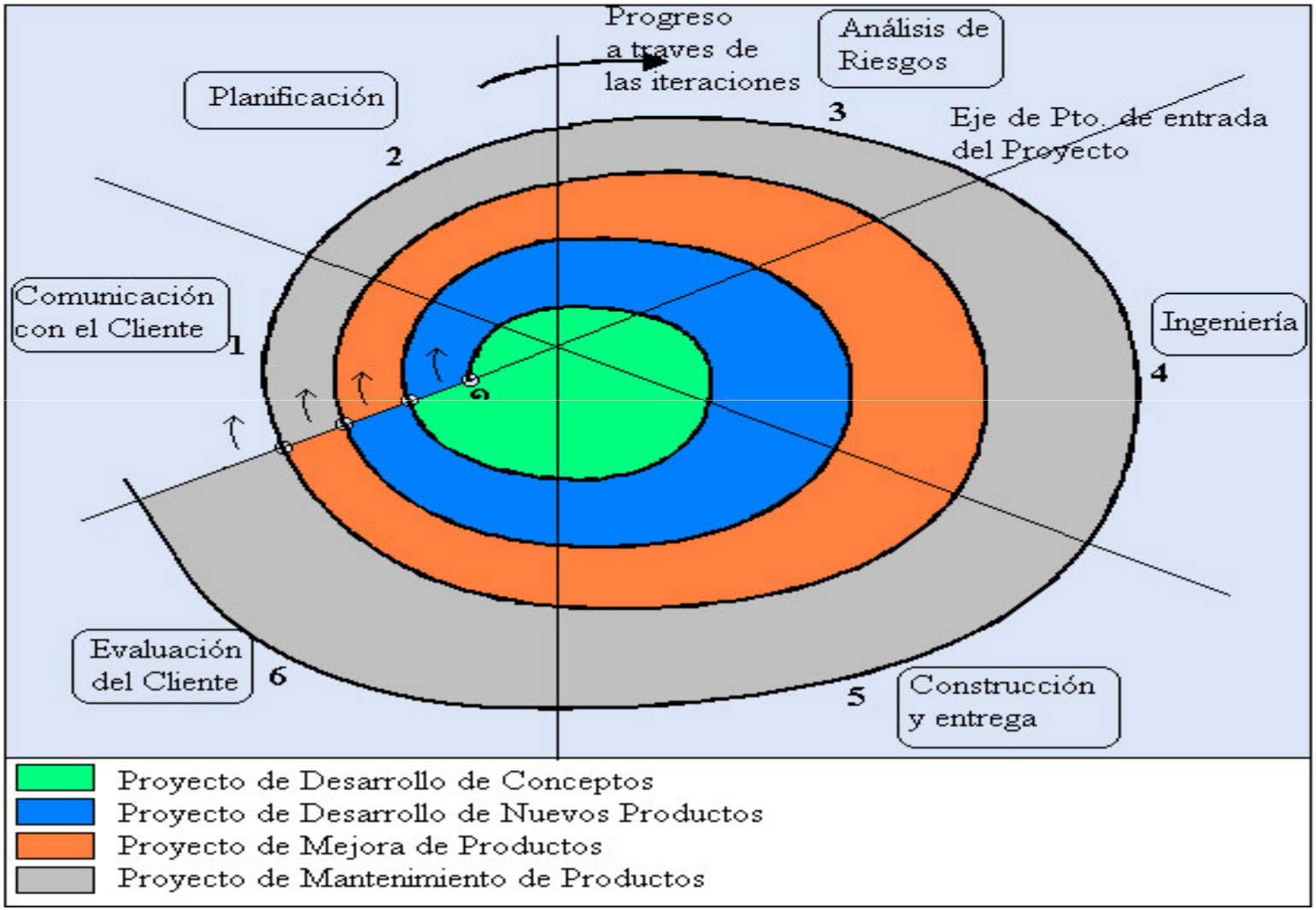
PARADIGMA 2: DESARROLLO EN ESPIRAL

- Se desarrolla un principio del Sistema, pero funcional en si.
- Se desarrolla luego una 2da parte, también funcional. Y acoplada a la primera.
- Esto se repite, para que en cada iteración, se obtenga una versión aumentada del sistema.

Comentarios:

- Integra lo mejor de los otros paradigmas.
- Tiene un alto costo de producción adicional.
- Tiene un punto de inflexión, que está en el análisis. Es complicado.

INGENIERIA DE SOFTWARE – Desarrollo en cascada



INGENIERIA DE SOFTWARE

PROBLEMAS

- **Suele aparecer una incapacidad para estimar el tiempo y el costo personal en el desarrollo de Software.**
- **Puede haber falta de calidad en el producto.**
- **Que nuestro método y producto no pueda adaptarse a los avances del Hw y la complejidad de los nuevos proyectos.**

Preguntas problemáticas:

- **Porque lleva tanto tiempo terminar los programas de Software?**
- **Por que es tan alto el costo del Sw?**
- **Porque no se pueden encontrar todos los errores antes de entregar terminado el producto ?**
- **Por que es tan difícil o imposible, controlar el progreso del desarrollo del Sw?**

INGENIERIA DE SOFTWARE

RETOS DEL SIGLO XXI

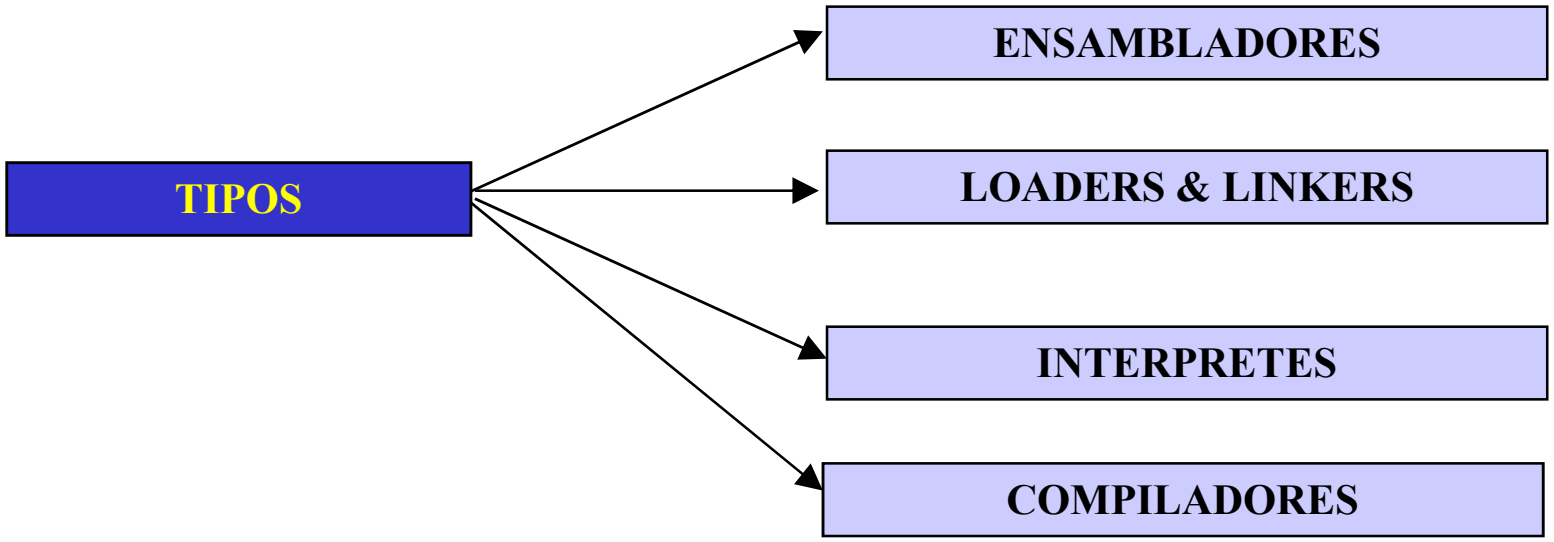
- Mantener lo heredado.
- Actualizar Sw antiguo con funciones críticas. Sin que deba parar la explotación y aumentar exponencialmente los costos.
- Que lo heterogéneo sea un ventaja
- Desarrollar sistemas flexibles y portables.
- Reducir tiempos de entrega, sin perjudicar la calidad del producto.

TIPOS DE SOFTWARE

1. MICROAPLICACIONES
2. SOFTWARE DE SISTEMA
3. SISTEMAS OPERATIVOS
4. SISTEMAS DE TIEMPO REAL
5. GESTION
6. CIENTIFICO & INGENIERIA
7. SUPERCOMPUTO.
8. SOFTWARE PARA PC
9. SOFTWARE EMBEBIDO
10. INTELIGENCIA ARTIFICIAL

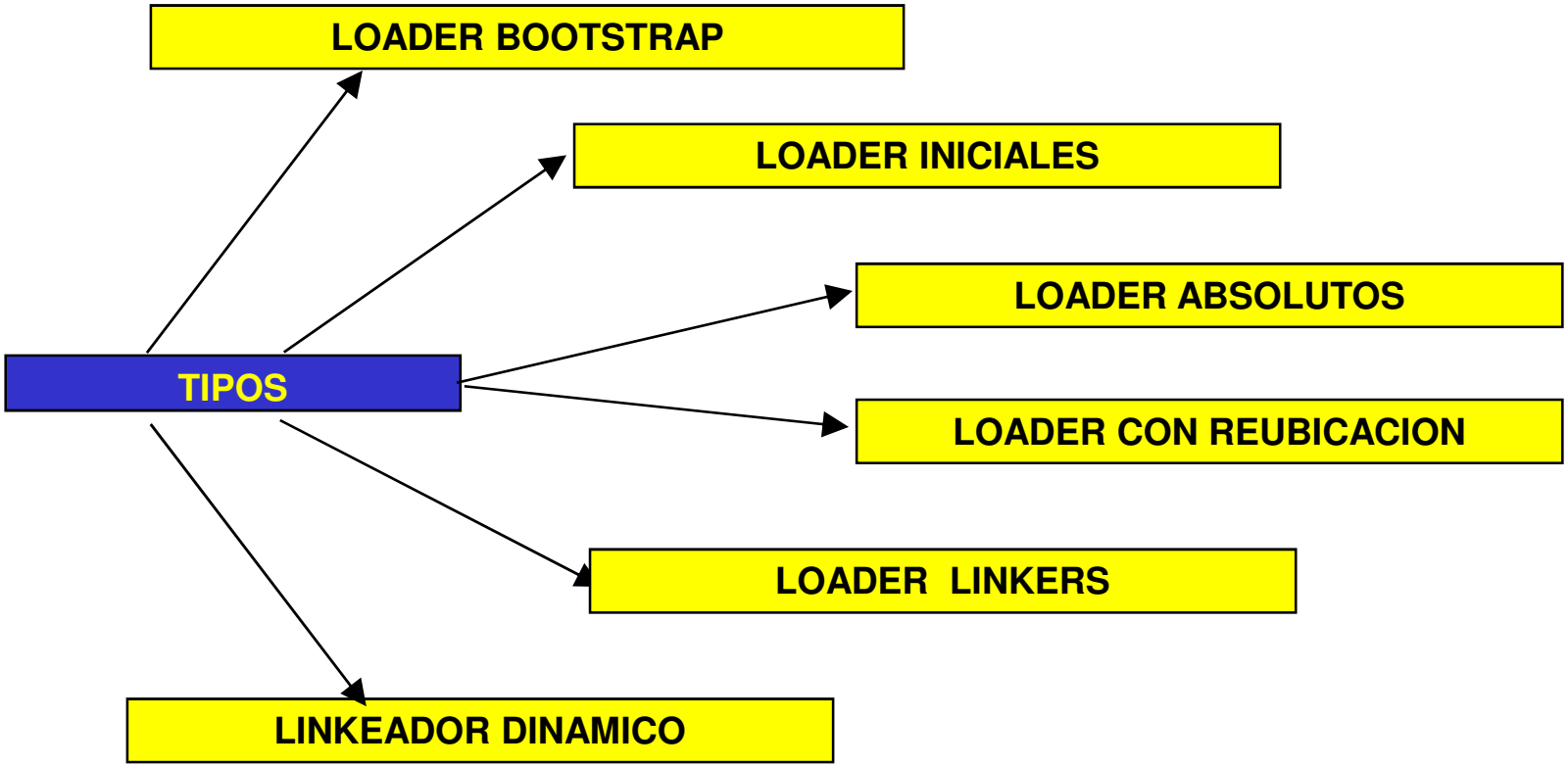
SOFTWARE DE SISTEMA

PROGRAMAS QUE CONFORMAN UTILERIAS DEL SISTEMA OPERATIVO,
QUE LE PERMITEN REALIZAR TAREAS INTERNAS DE OPERACION



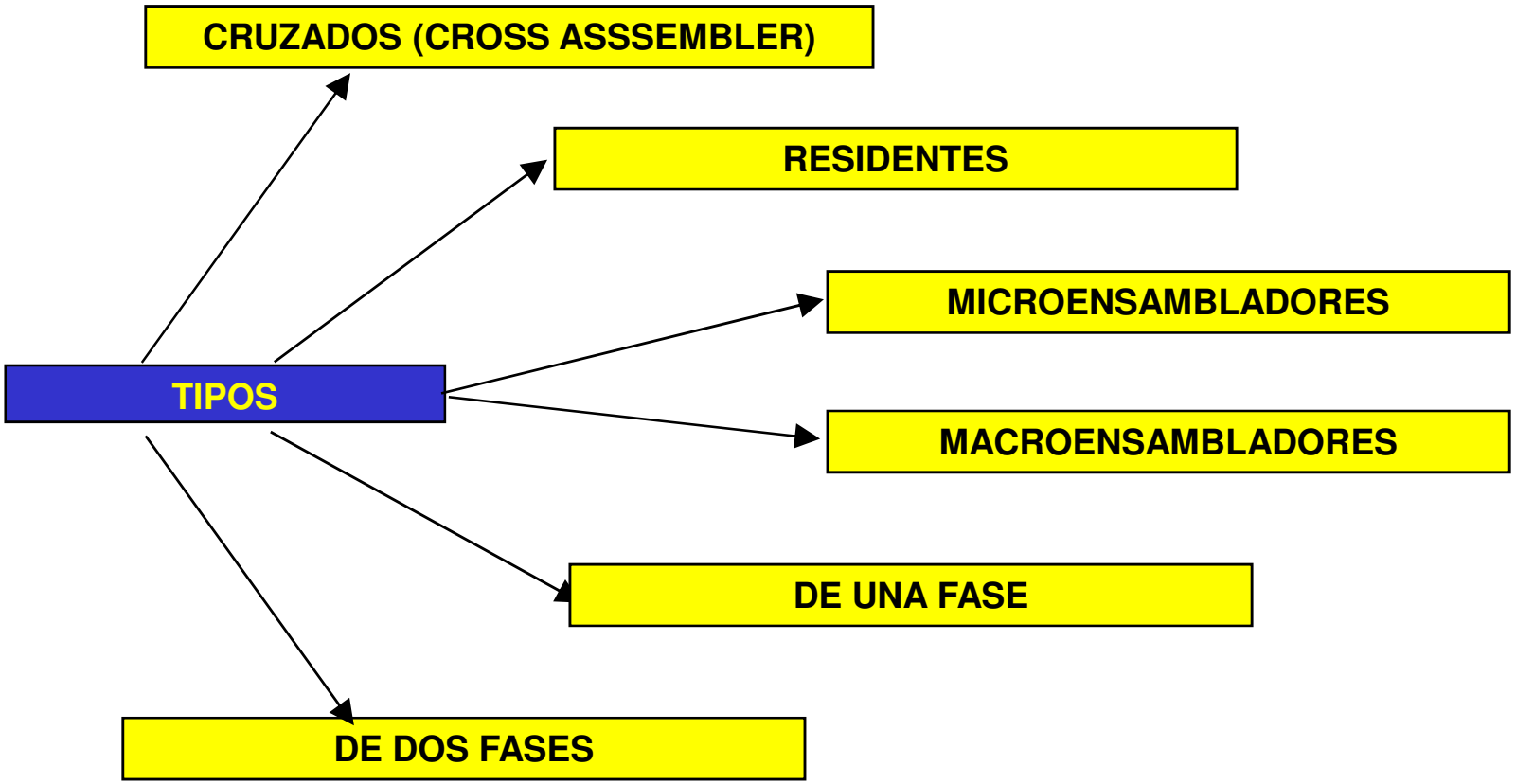
LOADERS & LINKERS

LOADER: PRG QUE COLOCA EN MEMORIA PARA SU EJECUCION A UN PRG DE USUARIO
LINKER: ENLAZA PRGs OBJETO PERMIITENDO REFERENCIAS CRUZADAS ENTRE VARIABLES Y CODIGO.



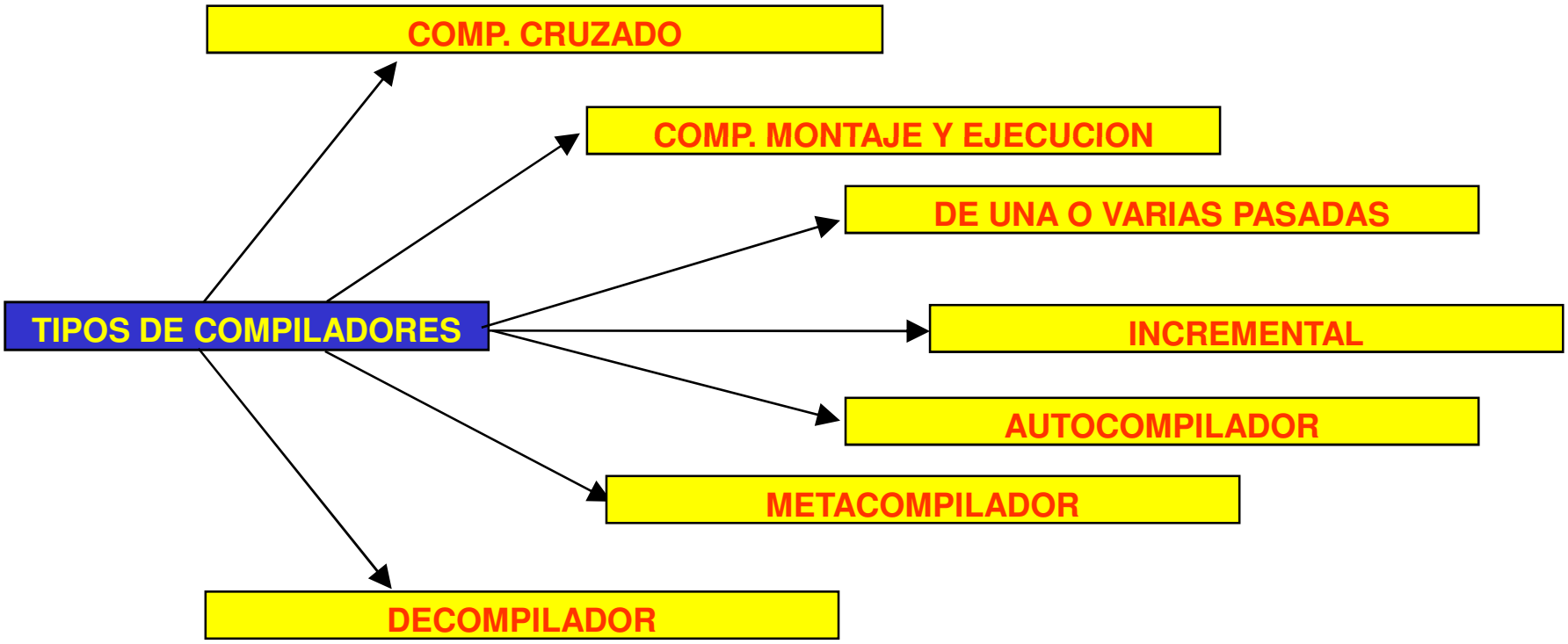
SOFTWARE DE SISTEMA: ENSAMBLADORES

TRADUCTOR DE PROGRAMAS EN LENGUAJE ASSEMBLER A CODIGO CORRESPONDIENTE AL LENGUAJE MAQUINA



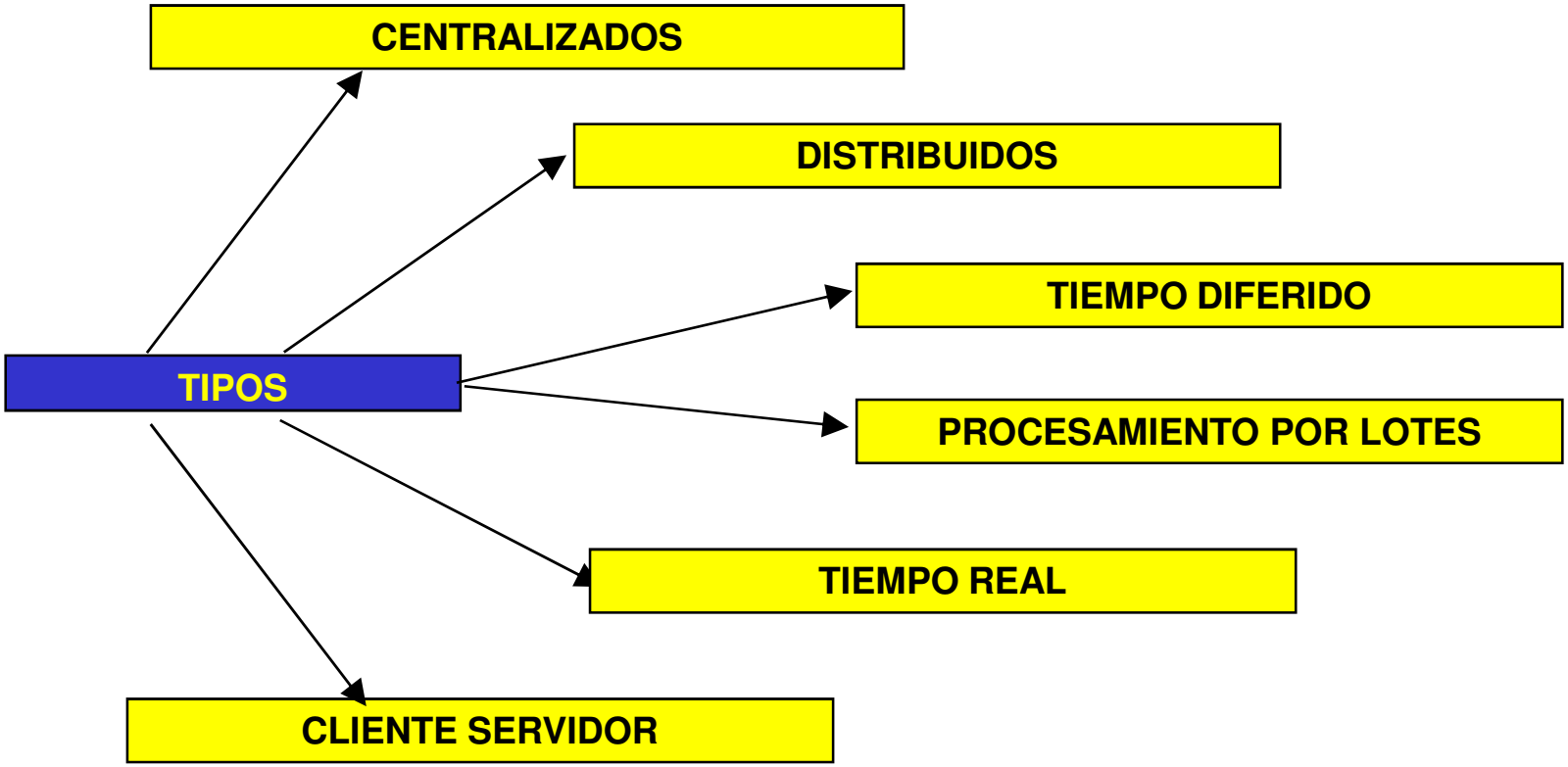
INTERPRETES Y COMPILADORES

INTERPRETES: TRADUCTOR QUE REALIZA LA EJECUCION
LINEA A LINEA DE UN CODIGO, CON FASES DE EDICION E EJECUCION
ESTAN INTEGRADAS.
COMPILADOR: PRG QUE CONVIERTE CODIGO FUENTE EN CODIGO
OBJETO, SE EJECUTA POR BLOQUE.



SOFTWARE DE SISTEMAS OPERATIVOS

GESTION Y COMUNICACIÓN DE UN SISTEMA COMPUTACIONAL. ADMINISTRACIÓN Y CONTROL.



SOFTWARE DE TIEMPO REAL

- ❖SISTEMAS DE CONTROL ADAPTIVO.
- ❖SISTEMAS DE ADQUISICIÓN Y ACTUACIÓN.
- ❖SISTEMAS DE EJECUCION REMOTA.

SOFTWARE DE GESTION

❖SISTEMAS DE ERP (Enterprice Resources Planning) –
Sistemas de Información Gerencial.
manejan la producción, logística, distribución, inventario, envíos, facturas y contabilidad de la compañía. Ej. OasisERP, W3ERP.

❖SISTEMAS CRM (Customer Relationship Manager) –
Sistema de Gestión orientado al Cliente.
Administra un Data Warehousing (Almacen de Datos), Gestión de Ventas y Relación con los Clientes.

LENGUAJES DE PROGRAMACION

- ALTO NIVEL . PASCAL – PYTHON - COBOL
- NIVEL INTERMEDIO. C
- BAJO NIVEL . ENSAMBLADOR
- LENGUAJE MÁQUINA. PSEUDOCÓDIGO BINARIO

COMPONENTES:

A. LOGICA
B. SINTAXIS.
C. INSTRUCCIONES: I/O – ARIT/LOG – SELECTIVAS – REPETITIVAS

LENGUAJES DE PROGRAMACION

TIPOS DE DATOS:

- SIMPLES: NUMERICOS – LOGICOS – CARÁCTER
- COMPLEJOS: ARREGLOS – ENUMERADOS – SUBRANGO – TYPE

CONSTANTES Y VARIABLES

EXPRESIONES ARITMETICAS

LENGUAJES DE PROGRAMACION

Tipos de Datos:

- Numéricos: Entero (Integer) Por Ej: -9, 0, 9
  Real (Real) Pej; -0.99 , 0.1, 3.01

Notación exponencial

36752010000000000000000

367 520 100 000 000 000 000

Potencias de 10

3.675201 x 10↑19

O también

0.000000000302579

O también

3.02579 x 10↑ -11 (coma flotante)

MANTISA

EXPONENTE

Todas son equivalentes:

3.675201x10↑19 = .3675201x10↑20=.03675201x10↑21

LENGUAJES DE PROGRAMACION

Tipos de Datos:

- ❖Lógicos: logicos boolean (True – False)
- ❖Carácter: Char – String (Alfanumérico y Caracteres Especiales)

Expresiones Aritméticas:

+, - , *, /, ↑, **, div (entera), mod (módulo (resto))

a div b Da parte entera Pej. 19div6=3
a mod b Da el resto Pej. 19mod6=1

LENGUAJES DE PROGRAMACION

Position Oct 2009	Position Oct 2008	Delta in Position	Programming Language	Ratings Oct 2009	Delta Oct 2008	Status
1	1		Java	18.718%	-2.23%	A
2	2		C	16.891%	+1.33%	A
3	5		PHP	10.390%	+1.78%	A
4	3		C++	9.911%	-1.04%	A
5	4		(Visual) Basic	8.729%	-1.08%	A
6	8		C#	4.433%	+0.67%	A
7	6		Python	3.914%	-0.65%	A
8	7		Perl	3.776%	-0.64%	A
9	11		JavaScript	3.033%	+0.36%	A
10	10		Ruby	2.458%	-0.40%	A
11	9		Delphi	2.140%	-1.15%	A
12	13		PL/SQL	1.020%	0.00%	A
13	49		Objective-C	0.902%	+0.82%	A--
14	14		SAS	0.805%	+0.21%	A
15	16		Pascal	0.669%	+0.15%	A-
16	20		ABAP	0.661%	+0.22%	A-
17	19		Lisp/Scheme	0.605%	+0.12%	B
18	22		MATLAB	0.577%	+0.17%	B
19	12		D	0.570%	-0.76%	B
20	15		Lua	0.527%	-0.02%	B

BIBLIOGRAFIA DE REFERENCIA

- · Estructuras y Diseño de Computadoras (La Interfaz hardware/Software). David. Patterson y John Hennessy. 4ta Edición. Ed. Reverte. Barcelona, 2011.
- Organización y Arquitectura de Computadores. Willams Stallings. Prentice-Hall. 2006. 7ed.
- · Organización y Arquitectura de Computadoras. Jaime Martinez Garza, Jorege Agustín Olvera Rodríguez. Prentice-Hall. 1era Edición. 2000.
- · Manual de Actualización y reparación de PCs, 12 edición. Scott Mueller. Que, Prentice Hall, 2001.
- · Organización de Computadores, un enfoque estructurado, 7 edición. Andrew Tanenbaun. Prentice Hall, 2001.
- · ESTRUCTURA INTERNA DE LA PC. Gastón C. Hillar. Ed. Hasa. 4ta. Edición. Bs.As.Feb. 2004.
- · ORGANIZACIÓN Y ARQUITECTURA DE COMPUTADORES. Willams Stallings. Prentice-Hall. 2000.
- · CIENCIAS DE LA COMPUTACION. Brookshear. Addison Wesley.
- · REDES DE ORDENADORES. Andrew Tannenbaum. Prentice Hall.

FIN DE LA MODULO IV SOFTWARE