



CAPITULO 1: INTRODUCCIÓN

Ing. Ventre, Luis O.

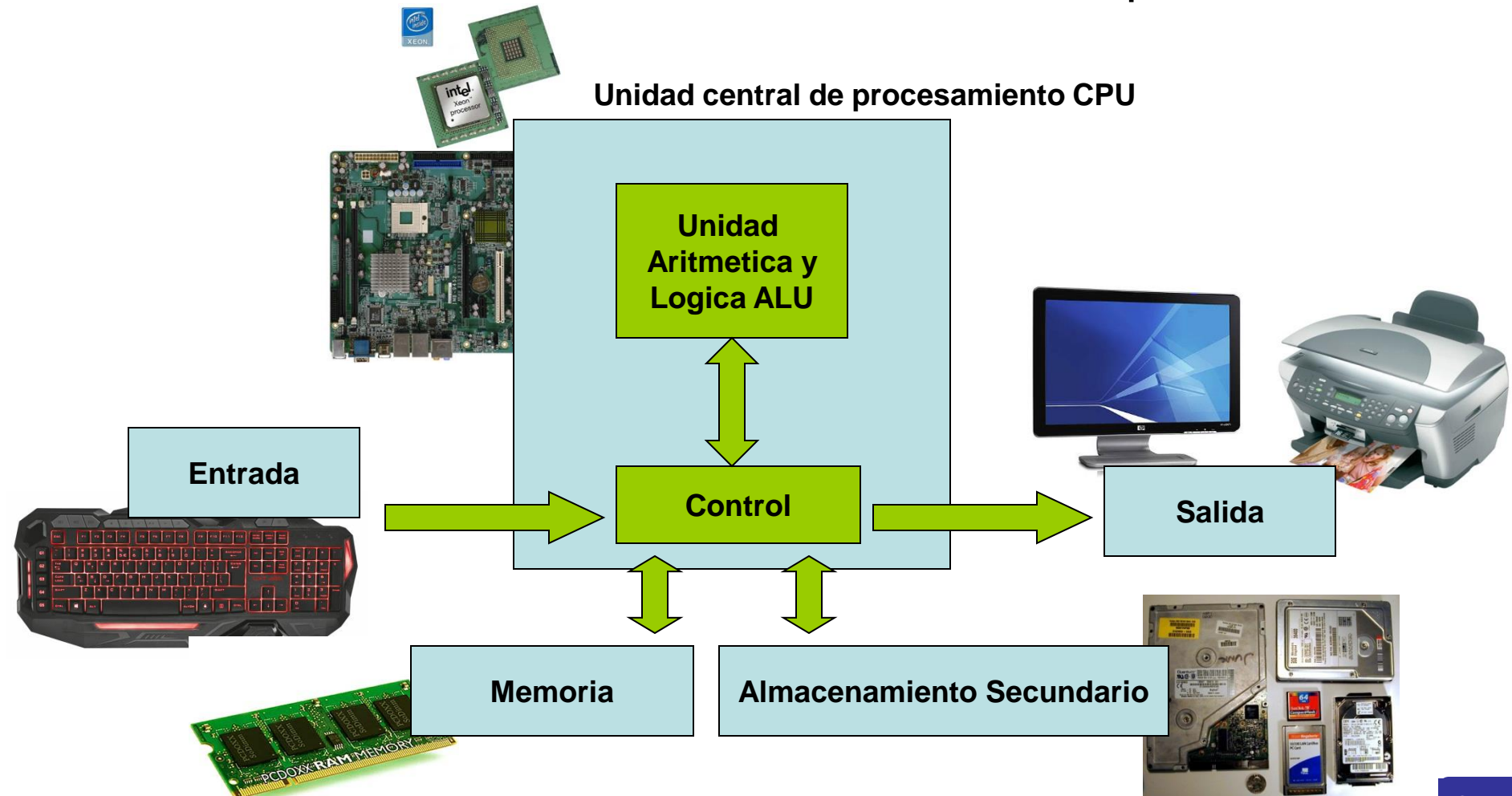


1.1 Computadores

- Computador, según RAE:
- «Máquina electrónica, analógica o digital dotada de una memoria de gran capacidad y de métodos de tratamiento de información, capaz de resolver problemas matemáticos y lógicos mediante la utilización automática de programas informáticos.»
- 2017 - Máquina electrónica capaz de realizar un tratamiento automático de la información y de resolver con gran rapidez problemas matemáticos y lógicos mediante programas informáticos

1.1 Computadores

- Unidades básicas de hardware de una computadora:





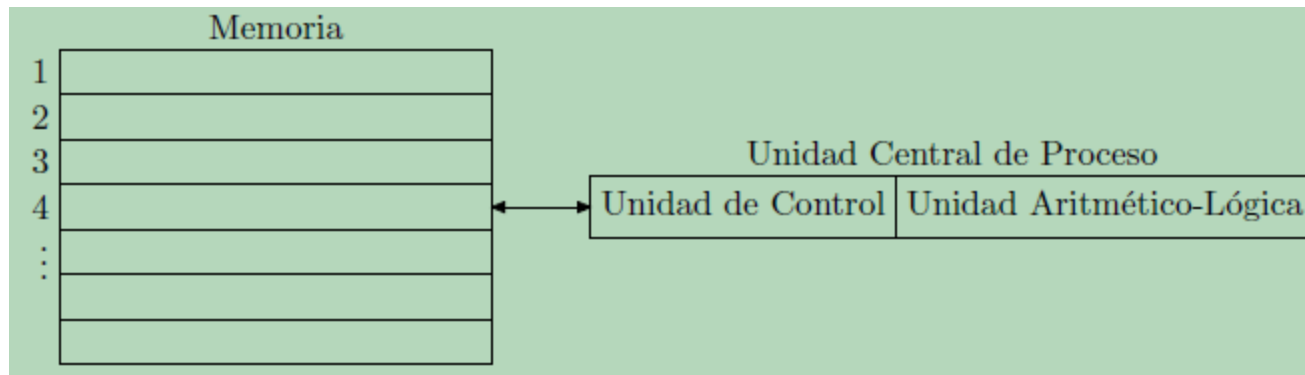
1.1 Computadores

- La unidad de memoria almacena la información en un formato lógico consistente. Esta almacena instrucciones y datos.
- La ALU, unidad aritmética y lógica, ejecuta todas las funciones aritméticas y lógicas como adición, sustracción, etc y las proporcionadas por la computadora.
- La unidad de control, dirige y supervisa la operación general de la computadora. Además controla la ejecución de todas las instrucciones.
- La UAL en conjunto con la Unidad de control, forma la unidad central de procesamiento o CPU.
- Armario y persona con calculadora



1.1 Computadores

- Memoria:
- Analogía: En vez de ser cajones de un armario, son celdas. Y cada celda tiene una dirección asociada.



- Cada posición de memoria permite almacenar una secuencia de unos y ceros de tamaño fijo. Dispositivos binarios.



- Almacenamiento secundario:
- **La memoria principal de la computadora es volátil.** Esto implica que cuando se quita la energía se pierde la información en ella. Por lo tanto no es útil para almacenamiento permanente de programas y datos.
- Para este propósito se usan **dispositivos de almacenamiento secundario** o auxiliar.
- Aunque en la antigüedad la información se haya guardado en tarjetas perforadas, cintas de papel, etc en la actualidad generalmente se almacena en pendrives, SSD, discos duros, dvds, etc.
 - Años 90 1 Diskette 3 1/2 1.44 Mb.
 - Año 2018 SSD 60 TB = 60000 GB = 41 millones de veces



1.2 Codificación

- Como se almacena la información en la memoria de un computador? Codificando...
- Codificar es asociar signos con los elementos de un conjunto.
- Los ordenadores disponen de solo dos signos básicos, $\{0, 1\}$; pero pueden combinarse en secuencia.
- Se almacenan en secuencias de unos y ceros, porque?...



1.2 Codificación

- La menor variable posible a almacenar en un computador se llama “**bit**” y puede tomar solo uno de 2 valores {0, 1}. Pero es habitual trabajar con secuencias de bits de números fijos.
- **Una secuencia de 8 bits recibe el nombre de “byte”.**



1.2 Codificación

- Cuantos significados pueden representarse con una secuencia de 8 bits?....

Es posible representar 256 significados, veamos el porque...

- Se puede representar el rango $[0, 255]$ con 8 bits. Debido a $2^8 = 256$.
- Para simplificar la programación, la correspondencia entre secuencias de bits y los números esta dada por la “**representación posicional**”.
- La cadena de bits: **b7b6b5b4b3b2b1** según representación posicional será: $\sum_{i=0}^7 b_i \cdot 2^i$



1.2 Codificación

- Veamos un ejemplo: la secuencia de bits 00001011

codifica el valor

00001011

$$\sum_{i=0}^7 b_i \cdot 2^i$$
$$0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 =$$
$$8 + 2 + 1 = 11$$

Diagram description: The binary sequence 00001011 is shown in bold. Eight arrows point from each bit to its corresponding term in the expansion of the sum of bits multiplied by powers of 2. The final result of the sum is 11.

- El bit mas a la izquierda es el **mas significativo** y el bit de mas a la derecha el **menos significativo**.
- Esta representación es adecuada para las operaciones, por ejemplo la suma, que posee una tabla como veremos ...



1.2 Codificación

- Tabla de sumar:

sumandos		suma	acarreo
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- El acarreo no nulo, demuestra que un bit para los resultados de la suma no alcanza, por lo tanto debe agregarse un segundo bit.
- Veremos a continuación una suma de dos secuencias de 8 bits



1.2 Codificación

- Sumaremos los valores 11 y 3 según su representación:

$$\begin{array}{r} 00001011 \\ + 00000011 \\ \hline \end{array}$$

comenzando por los bits menos significativos:

$$\begin{array}{r} \text{Acarreo} \qquad \qquad 1 \\ 00001011 \\ + 00000011 \\ \hline 0 \end{array}$$

en el siguiente paso, 1 mas 1 es 0, mas el acarreo 1:

$$\begin{array}{r} \text{Acarreo} \qquad \qquad 11 \\ 00001011 \\ + 00000011 \\ \hline 10 \end{array}$$



1.2 Codificación

De la forma antes mencionada llegamos al resultado:

$$\begin{array}{r} \text{Acarreo} \qquad \qquad \qquad 1 \ 1 \\ \qquad \qquad \qquad 00001011 \\ + \qquad \qquad \qquad 00000011 \\ \hline \qquad \qquad \qquad 00001110 \end{array}$$

lo cual lógicamente es el numero 14.

- Para la representación de números negativos, existen 3 formas, la intuitiva utiliza el bit mas significativo, 0 en caso positivo y 1 en caso negativo, esto limita el rango de valores a [-128, 127]. Y es compleja su implementación.
- También pueden representarse números negativos mediante “complemento a uno” y “complemento a dos”.



1.2 Codificación

- **Complemento a uno**, para representación de números negativos. Se toma la representación posicional del numero (debe poder expresarse con 7 bits) y se invierte todos sus bits.
- Se realiza la suma convencional y si se produce un desbordamiento debe sumarse uno al resultado.

Veamos un ejemplo de la suma de 3 y -2

El numero 2 es la secuencia 00000010
 el cual con todos sus bits invertidos es
 11111101

En caso de existir acarreo son necesarias
 dos sumas! poco efectivo...

$$\begin{array}{r}
 \text{Acarreo} \quad 1111111 \\
 00000011 \\
 + 11111101 \\
 \hline
 (1)00000000 \\
 \downarrow \\
 00000000 \\
 + 00000001 \\
 \hline
 00000001
 \end{array}$$



1.2 Codificación

- **Complemento a dos**, utilizada por los ordenadores, mas efectiva, aunque es poco natural.
- Se debe invertir todos los bits de la representación del numero negativo y al resultado sumarle 1.
- El mismo ejemplo de 3 y -2 seria:

El numero 2 es la secuencia 00000010
 el cual con todos sus bits invertidos es
 11111101 y se le debe sumar 1 =
 11111110.

Acarreo	1 1 1 1 1 1	
	00000011	
+	11111110	→

	(1)00000001	

Para el caso anterior son necesarias 2 sumas, para este solo una!



1.2 Codificación

- Y como representar letras con esta codificación?
- Existe una tabla que pone en correspondencia 127 signos con secuencias de bits y es “asumida” como estándar, es denominada **Tabla ASCII**

Dec	Chr	Dec	Chr	Dec	Chr
32	Space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(72	H	104	h
41)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p



1.3 Programas y Lenguajes de Programación

- Decíamos que la memoria era un almacén con cajones, con secuencias de bits. Estos representan datos e instrucciones.....
- Que es un programa?
- Conjunto independiente de instrucciones usado para operar una computadora
- Cual es el objetivo de un programa?
- Producir un resultado específico a través de la computadora.
- Sinónimos de programas o conjuntos de prog. es Software.



1.3 Programas y Lenguajes de Programación

- Que es la programación?
- Es el proceso de escritura de las sentencias o instrucciones que componen un programa.
- Que es el lenguaje de programación?
- Cualquier sistema de notación que permite expresar programas, con un grupo de recursos semánticos.
- Existen diferentes formas y tipos de lenguajes de programación.



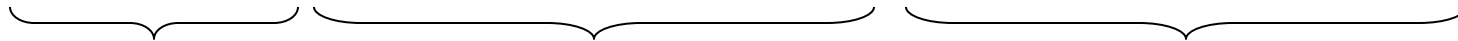
1.3 Programas y Lenguajes de Programación

- **Codificación de Maquina**
- En el nivel mas fundamental los únicos programas que pueden ejecutarse para operar una computadora son los programas en lenguaje de maquina. Consisten en una secuencia de instrucciones compuestas por **números binarios** como:
 - 11000000 000000000001 000000000010
 - Estas instrucciones están conformadas por dos partes, una de instrucción y dos de dirección. Que hacer y con quienes?
 - Cada maquina tiene su codificación particular.



1.3 Programas y Lenguajes de Programación

• 11000000 000000000001 000000000010



• Opcode Dirección Dato 1 Dirección Dato 2

• Veamos un ejemplo como sería en código de maquina:

Memoria

1	Sumar contenido de direcciones 10 y 11 y dejar resultado en dirección 13
2	Sumar contenido de direcciones 13 y 12 y dejar resultado en dirección 13
3	Dividir contenido de dirección 13 por 3 y dejar resultado en dirección 13
4	Detener

Memoria

1	10101011 00001010 00001011 00001101
2	10101011 00001101 00001100 00001101
3	00001110 00001101 00000011 00001101
4	00000000 00000000 00000000 00000000
⋮	

Unidad Central de Proceso (CPU)





1.3 Programas y Lenguajes de Programación

- Y la ejecución procedería....

Memoria

1	Sumar contenido de direcciones 10 y 11 y dejar resultado en dirección 13
2	Sumar contenido de direcciones 13 y 12 y dejar resultado en dirección 13
3	Dividir contenido de dirección 13 por 3 y dejar resultado en dirección 13
4	Detener
:	:
10	5
11	10
12	6
:	:



1.3 Programas y Lenguajes de Programación

- Y la ejecución procedería....

Memoria

1	Sumar contenido de direcciones 10 y 11 y dejar resultado en dirección 13
2	Sumar contenido de direcciones 13 y 12 y dejar resultado en dirección 13
3	Dividir contenido de dirección 13 por 3 y dejar resultado en dirección 13
4	Detener
:	:
10	5
11	10
12	6
13	15
:	:



1.3 Programas y Lenguajes de Programación

- Y la ejecución procedería....

Memoria

1	Sumar contenido de direcciones 10 y 11 y dejar resultado en dirección 13
2	Sumar contenido de direcciones 13 y 12 y dejar resultado en dirección 13
3	Dividir contenido de dirección 13 por 3 y dejar resultado en dirección 13
4	Detener
:	:
10	5
11	10
12	6
13	21
:	:



1.3 Programas y Lenguajes de Programación

- Y la ejecución finalizaría....

Memoria

1	Sumar contenido de direcciones 10 y 11 y dejar resultado en dirección 13
2	Sumar contenido de direcciones 13 y 12 y dejar resultado en dirección 13
3	Dividir contenido de dirección 13 por 3 y dejar resultado en dirección 13
4	Detener
⋮	⋮
10	5
11	10
12	6
13	7
⋮	⋮



1.3 Programas y Lenguajes de Programación

- **Lenguajes Ensambladores**
- Programar en lenguaje de maquina es tedioso y lento. En un pasado lejano se hacia con cables en un panel de conectores.

- Reemplazo de:

opcodes binarios
direcciones



**Mnemotecnia/palabras
Etiquetas**

- El programa traductor a lenguaje maquina es el
ENSAMBLADOR

```
SUM #10, #11, #13  
SUM #13, #12, #13  
DIV #13, 3, #13  
FIN
```



1.3 Programas y Lenguajes de Programación

- **Lenguajes Ensambladores**
- Utilizar # antes de la posición de memoria implica el contenido de la misma.
- El conjunto de códigos mnemotécnicos traducible a código de maquina la reglas para combinarlos, expresar direcciones, etc recibe el nombre de lenguaje ensamblador.

```
SUM #10, #11, #13  
SUM #13, #12, #13  
DIV #13, 3, #13  
FIN
```



1.3 Programas y Lenguajes de Programación

- **Cada ordenador, tiene asociado un grupo de posibles instrucciones de acuerdo a su estructura física, por lo tanto también un lenguaje ensamblador...**
- **Para cada ordenador, un programa distinto?**
- **Hay una posible solución?**
- **Un lenguaje ensamblador universal?**





1.3 Programas y Lenguajes de Programación

- **Lenguajes de bajo nivel:**
- Lenguajes de maquina y ensambladores, son de nivel bajo.
- Rápida Ejecución.
- Complejos.
- Particular al tipo de computadora.

- **Solución intermedia: Lenguajes de alto nivel:**
- Instrucciones similares a lenguajes escritos.
- Multi plataforma.
- Python, C, C++, java, etc.



```
a = 5
b = 10
c = 6
media = (a + b + c) / 3
```



1.3 Programas y Lenguajes de Programación

- Los programas escritos en un lenguaje de computadora se conocen como CODIGO FUENTE.
- Los lenguajes de alto nivel también **deben traducirse a lenguaje máquina**. Y existen 2 formas:
 - Cada declaración se traduce y ejecuta individualmente ***lenguaje Interpretado – Interprete***
 - Todas las instrucciones se traducen como una unidad completa antes de ejecutar ninguna ***lenguaje Compilado – Compilador***.
- Python lenguaje interpretado!..C++ lenguaje compilado.



1.3 Programas y Lenguajes de Programación

- Porque PYTHON?...
- Muy EXPRESIVO, programas compactos....
- Legibilidad...sintaxis de lectura simple.
- Entorno interactivo facilita la tarea.
- Gratuidad de su interprete!...
- Puede ser usado como lenguaje procedural, rico en estructuras etc...



1.3 Programas y Lenguajes de Programación

- En concreto:

¡Hola de nuevo, mundo!

Te presentamos los programas «¡Hola, mundo!» en Python (izquierda) y C (derecha).

```
print 'Hello, world!'
```

```
#include <stdio.h>

int main(void) {
    printf("Hello, world!\n");
    return 0;
}
```

Como puedes comprobar, Python parece ir directamente al problema: una sola línea. Empezaremos aprendiendo Python.



1.3 Programas y Lenguajes de Programación

- Y C?
- Es de propósito general...
- Uno de los mas utilizados en el mundo profesional...
- Permite controlar con precisión los factores que influyen en la eficiencia de los programas, con el alto precio de la escritura mas **fuertemente tipada y ciertas complicaciones semanticas.**



1.4 Mas allá de los programas: Algoritmos

- **Algoritmo: Que es?**
- El procedimiento seleccionado para producir el resultado de un problema.
- Mas precisamente es una **secuencia paso a paso de instrucciones que deben realizarse para la consecución de un objetivo y describe como han de procesarse los datos para producir las salidas deseadas.**



1.4 Mas allá de los programas: Algoritmos

- **Algoritmo: Que es?**
- En esencia responde a la pregunta
Que método se usara para resolver este problema?

Desde esta definición: la programación es la traducción de un algoritmo seleccionado a un lenguaje que puede usar la computadora.



1.4 Mas allá de los programas: Algoritmos

- Una computadora es una maquina que responde únicamente a algoritmos, esto equivale a decir que responde a una **secuencia detallada de instrucciones paso por paso.**
- Un algoritmo puede describirse de varias formas:
- Cuando se utiliza el lenguaje natural para describir el algoritmo la descripción se llama **pseudocódigo.**
- Cuando se utilizan ecuaciones matemáticas la descripción se llama **fórmula.**
- Cuando se utilizan simbología grafica se la descripción se llama **diagrama de flujo.**
- Una vez lista la descripción se codifica el pseudocódigo



1.4 Mas allá de los programas: Algoritmos

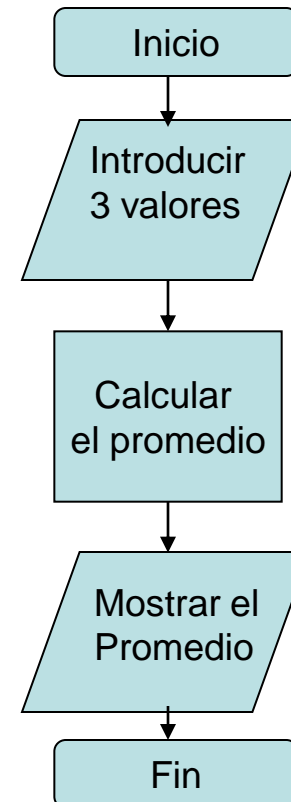
- Ejemplos:
- Suponga un programa cuyo objetivo es obtener el promedio de tres números:
- **Pseudocódigo:**
 - Solicitar la introducción del primer elemento.*
 - Solicitar la introducción del segundo elemento.*
 - Solicitar la introducción del tercer elemento.*
 - Calcular el promedio sumandos los números y dividiendo en 3.*
 - Mostrar el promedio*
- **Fórmula**
 - Establezca a,b,c igual a 3 valores*
 - Establezca $sum=a+b+c$*
 - Establezca $promedio=sum/3$*
 - Imprima promedio*



1.4 Mas allá de los programas: Algoritmos

- Ejemplos:
- Suponga un programa cuyo objetivo es obtener el promedio de tres números:

- **Diagrama de flujo**





1.4 Mas allá de los programas: Algoritmos

- Algoritmo:
 1. Ha de tener 0 o mas datos de entrada.
 2. Debe proporcionar 1 o mas datos de salida.
 3. Cada paso, exacto y sin ambigüedad.
 4. Debe ser finito, llegar a la solución con un determinado numero de pasos. Cada paso debe ser ejecutable en tiempo finito.
 5. Efectivo, sus paso no solo deben ser solubles en tiempo finito, sino que también con recursos determinados.

Y si puede ser, también eficientes, que alcancen el objetivo en el menor tiempo posible.



Siguientes filminas a título informativo



Errores comunes de programación

- De acuerdo a los temas vistos, es de importancia destacar los siguientes errores característicos en la programación:
- **Apresurarse a escribir y ejecutar el programa antes de entender por completo que se requiere,** incluyendo los algoritmos que se usaran para producir el resultado.
- Otro error típico es no respaldar el programa, normalmente ocurre, hasta que se produce una perdida importante.



A título informativo

- Las primeras computadoras en los años 50 eran grandes por los relés, tubos catódicos y por almacenamiento en tarjetas perforadas. Luego en los 60 con los transistores se combinó: la ALU, con la unidad de control dando lugar a la unidad central de procesamiento CPU. Esto proporcionó una velocidad de procesamiento mejorada.
- Posteriormente nace el IC, circuito integrado, 100 transistores en 1 cm².
- Las versiones actuales de estos IC contienen entre 0.7 y 2,27 billones de transistores Core i7 2da Gen y se conocen como chips integrados a gran escala.
- https://en.wikipedia.org/wiki/List_of_Intel_Core_i7_microprocessors



Hardware de computación y Conceptos de almacenamiento

- Palabras y direcciones:
- Uno o mas bytes pueden agruparse en unidades mas grandes llamadas palabras. Esto facilita un acceso mas rápido y mas extenso a los datos.
- Una recuperación de un dato con ancho de palabra de cuatro bytes es mas rápido que cuatro recuperaciones individuales.
- El ancho de palabra determina el numero máximo y mínimo que puede ser representado por esa palabra

Tamaño de palabra	Valor del numero entero máximo	Valor del numero entero mínimo
1 Byte	127	-128
2 Bytes	32767	-32768
4 Bytes	2147483647	-2147483648



1.1 Introducción a la programación

- **Software de aplicación**

Grupo de programas escritos para realizar tareas particulares requeridas por los usuarios.

- **Software de sistema**

Colección de programas que deben estar disponibles en cualquier sistema de computo en el que ha de operar.

Cargador inicial, bootstrap componente del soft de sistema.

De manera colectiva el conjunto de programas de sistema usados para operar y controlar una computadora se denomina **Sistema operativo**.



1.2 Solución y desarrollo de software

- Recordemos: Un programa es una solución desarrollada para resolver un problema particular. Por lo tanto escribir un programa **casi es el ultimo paso** en un proceso de desarrollo.
- El método usado por los profesionales que desarrollan software para entender el problema que se va a a solucionar y para crear una solución efectiva y apropiada se llama ***procedimiento de desarrollo de software***, y consiste de 3 fases que se superponen:
 - *Diseño y desarrollo*
 - *Documentación*
 - *Mantenimiento*



1.2 Solución y desarrollo de software

- Como disciplina la **ingeniería de software** se encarga de crear programas y sistemas legibles utilizando el **procedimiento de desarrollo de software**.
- **Fase I: Desarrollo y diseño.**
Esta fase consta de 4 pasos. Inicia con el planteamiento del problema, requerimiento de un programa.

Análisis – Diseño – Codificación – Testing o Pruebas

Analizar el problema: importante definición y entender claramente el problema. **Que salidas y entradas se necesitan.**

Primer paso y mas importante entender los requerimientos.



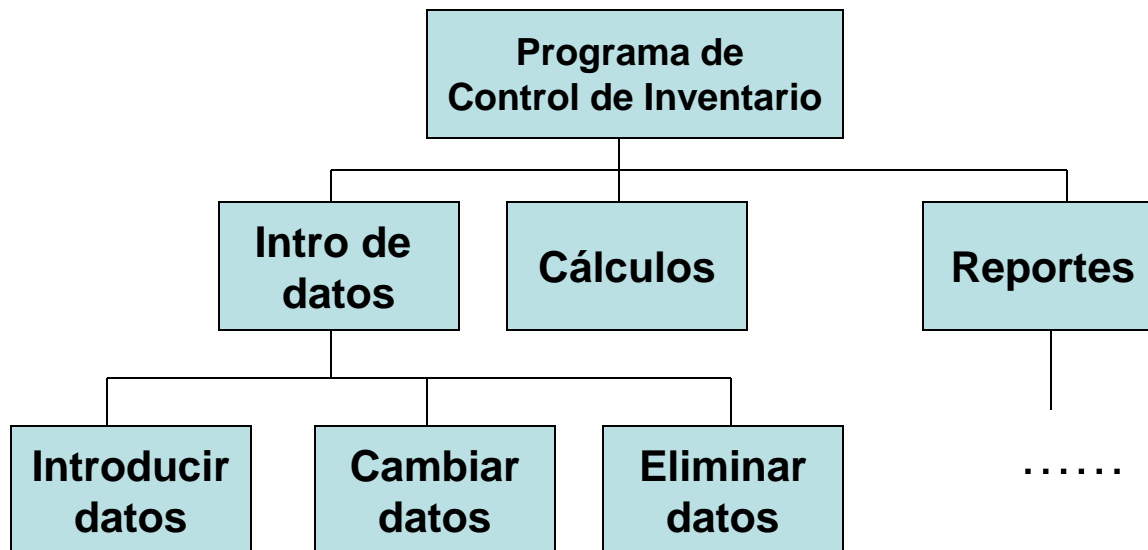
1.2 Solución y desarrollo de software

- Al finalizar la etapa de análisis se debe tener claro:
- Que debe hacer el sistema o programa?
- Que salidas debe producir?
- Que entradas requiere para crear dichas salidas?
- **Diseño de una solución.**
- Selección del conjunto detallado de pasos “algoritmo”, para resolver el problema. Obviamente será una etapa de **refinamiento** del algoritmo. Este algoritmo debe verificarse
Para refinar la solución pueden utilizarse diagramas de estructura.
En sistemas complejos la solución debe organizarse en **subsistemas mas pequeños.**



1.2 Solución y desarrollo de software

- En un **diagrama de estructuras** una vez definida una estructura inicial se refina hasta que todas las tareas estén definidas por completo, desde los requisitos de nivel mas alto hasta el menor.





1.2 Solución y desarrollo de software

- Una vez que todos los requerimientos están contemplados en el grafo, se debe **codificar la solución**, también conocido como escribir el programa.
- En un programa bien formado, su **estructura** responderá a los siguientes patrones:
 - Secuencia – Define el orden de ejecución de instrucciones.
 - Selección – capacidad para realizar elección.
 - Iteración – bucle, o repetición, capacidad de iterar.
 - Invocación – implica solicitar un conjunto de instrucciones.
- Una vez finalizada la codificación, se debe probar y corregir el programa. Verificar que el programa cumple su objetivo.



1.2 Solución y desarrollo de software

- En la practica verificar un problema requeriría comprobar **todas las combinaciones** posibles de ejecuciones de las instrucciones. Esto es por lo general imposible.
- Test de unidades y test de integración
- Si durante una prueba se encuentra un error(bug), puede iniciarse el proceso de **depuración**, el cual incluye *localizar, corregir y verificar corrección*.
- Para detectar y corregir errores en un programa es importante desarrollar un **conjunto de datos de prueba** por medio de los cuales determinar si el programa proporciona respuestas correctas.



1.2 Solución y desarrollo de software

- Esfuerzo promedio dedicado en la fase I:

Paso	Esfuerzo
Analizar el problema	10%
Desarrollar una solución	20%
Codificar la solución	20%
Probar el programa	50%



1.2 Solución y desarrollo de software

- **Fase II: Documentación**
- Documentar el trabajo es uno de las claves del éxito de un diseño. Mucho trabajo se pierde debido a una mala documentación. Se debe recopilar la información a lo largo de las etapas del desarrollo.

En esencia existen 5 documentos para toda solución de problema:

Descripción del problema.

Desarrollo y cambios del algoritmo

Listado del programa bien comentado

Muestras de las pruebas efectuadas

Manual del usuario



1.2 Solución y desarrollo de software

- **Fase III: Mantenimiento**
- Esta fase está relacionada con la corrección continua de problemas, revisiones para satisfacer *necesidades cambiantes* y la adición de *características nuevas*.
- Generalmente es la fuente principal de ingresos.
- Etapa **mas larga** del desarrollo
- Entre mas completa es la fase II, el mantenimiento es mas eficiente.
- **Respaldo.** Aunque no es parte del proceso de diseño formal, es esencial hacer y conservar copias de respaldo.
- Recuperación de la ultima etapa con esfuerzo mínimo.



1.2 Solución y desarrollo de software

- **Modelo de desarrollo de software antiguo.**

Etapas Modelo Cascada

No se superponen las etapas

Tiempo necesario prolongado

Cualquier cambio en requerimientos invalida el modelo



El proceso de desarrollo de software



1.2 Solución y desarrollo de software

- **Modelo de desarrollo de software en espiral B.Bohem**

***Cascada para cada etapa
No se define el sistema completo
Inicialmente.***

***Requerimientos mas importante
primero***

***En cada vuelta tenemos un
programa funcionando y
entregable.***

***Tolero mejor cambios en
requisitos***

