CAPITULO 1
INTRODUCCION A LA PROGRAMACION

Ing. Luis O. Ventre

1.1 Introducción a la programación

- Que es un programa?
- Conjunto independiente de instrucciones usado para operar una computadora

- Cual es el objetivo de un programa?
- Producir un resultado especifico através de la computadora.

- Otro termino utilizado para los programas o conjuntos de programas es Software.

1.1 Introducción a la 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?
- Es el conjunto de recursos semánticos utilizados para la programación.

- Existen diferentes formas y tipos de lenguajes de programación.

- **Lenguaje 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?

- 11000000 000000000001 000000000010

- Opcode Dirección Dato 1 Dirección Dato 2

- **Lenguajes Ensambladores**
- Programar en lenguaje de maquina es tedioso y lento. Reemplazo de opcodes binarios por **símbolos en forma de palabras y etiquetas en las direcciones**.

- SUMAR 1,2

- El programa traductor a lenguaje maquina es el **ensamblador**

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

- **Lenguajes de alto nivel:**
- **I**nstrucciones similares a lenguajes escritos.
- Multi plataforma.
- Visual Basic, C, C++, java, etc.
- EJ: *Resultado = (primero+segundo) * tercero*

1.1 Introducción a la 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:

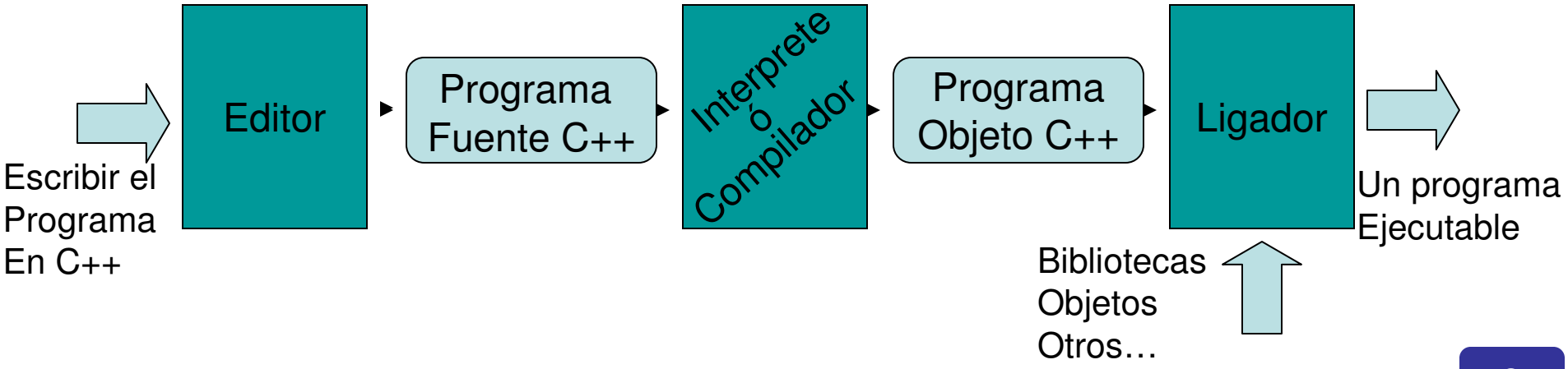
- Cada declaración se traduce y ejecuta individualmente ***lenguaje Interpretado – Interprete***
- Todas las instrucciones se traducen como una unidad completa antes de ejecutar ninguna

  lenguaje Compilado – Compilador.

- C++ lenguaje compilado.

- Ambiente de desarrollo. IDE

Aplicación de software que provee facilidades al programador para desarrollar software. Compuesto normalmente por un **editor de código fuente, un compilador o interprete, herramientas de automatización y debugger.**



- Orientación a **procedimientos**

  Lenguajes en donde las instrucciones disponibles se usan para crear **unidades independientes**, conocidas como procedimientos. Adquieren datos de entrada procesan y producen datos de salida.

- Orientación a **objetos**

  Lenguajes que deben definir los objetos que utilizaran, incluyendo una descripción general y unidades especificas para manipularlos. Reutilización del código existente con mayor facilidad, elimina necesidad de revalidar y reexaminar código nuevo o modificado.

- **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**.

1.1 Introducción a la programación

- Propósito de todo programa de aplicación: procesar datos para producir uno o mas resultados específicos.
- Lenguajes de procedimiento:
- **FORTRAN** (Formula translation)
- **COBOL** (Common Business Oriented Languaje)
- **BASIC** (Beginners All purpose Symbolic Instruction Code)
  No estructurado, difícil comprender luego poco tiempo.
- **PASCAL** para entender y reutilizar.
- ***Lenguaje C*** lenguaje de procedimientos estructurado. Ventaja escritura como lenguaje de alto nivel, mientras conserva capacidades de acceso directo a características del nivel de maquina.
- C++ conserva todo c, y agrega orientación a objetos.

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*

- 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.

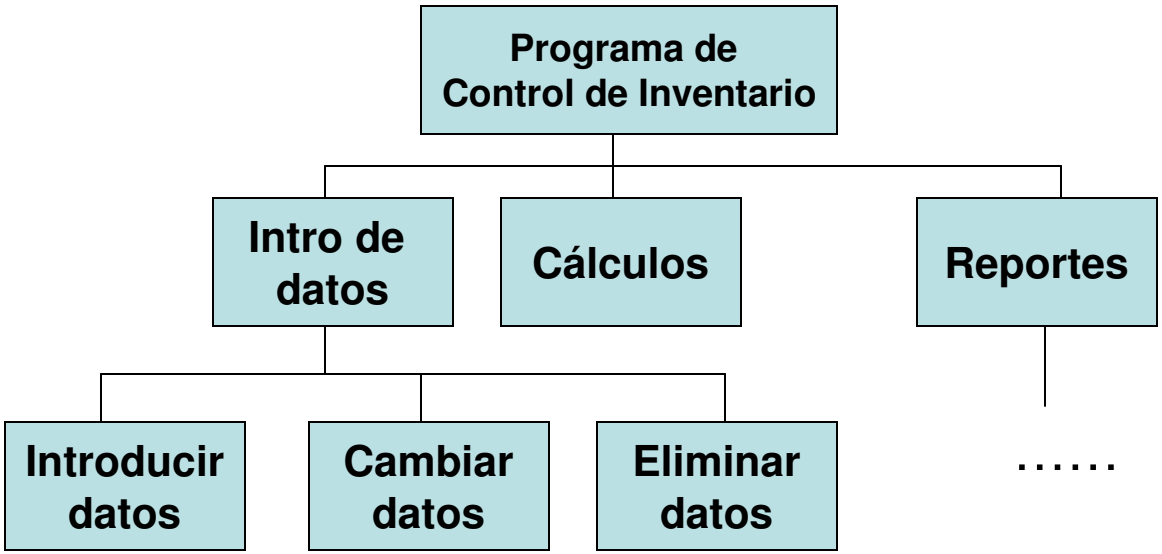
- Al finalizar la etapa de análisis se debe tener claro:
- Que debe hacer el sistema o programa?
- Que salidas debe producir?
- Que entradas requiere para crear dichas salidas?

- **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**.

- 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.

- 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.

- 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%

- **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

- **Fase III: Mantenimiento**
- Esta fase esta 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.

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

No se superponen las etapas

Tiempo necesario prolongado

Cualquier cambio en requerimientos invalida el modelo

El proceso de desarrollo de software

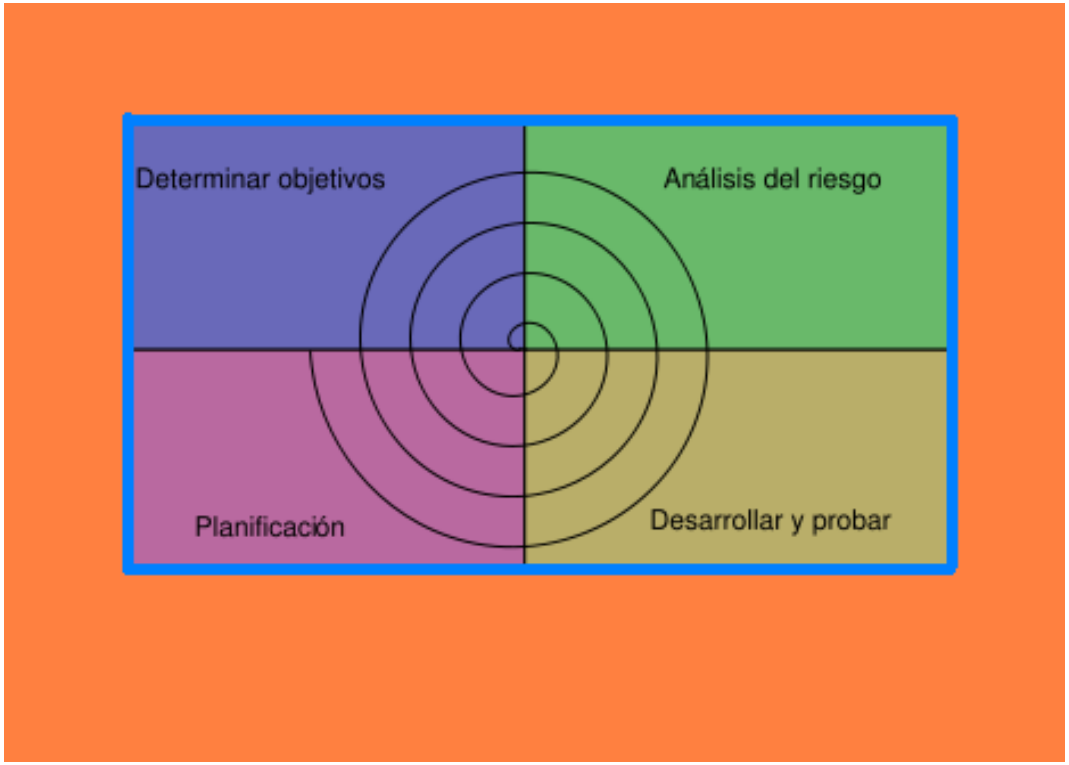
- **Modelo de desarrollo de software en espiral B.Bohem**

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



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

1.3 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 utilizan enunciados 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

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

- **Pseudocódigo:**

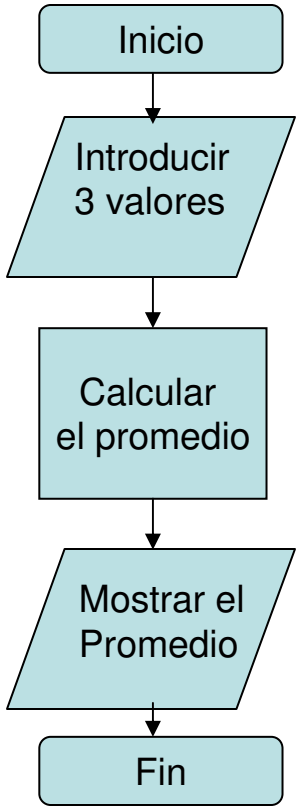
  Solicitar la introducción del primer elemento.
  Solicitar la introducción del segundo elemento.
  Solicitar la introducción del tercer elemento.
  Calcular el promedio sumandos los números y dividiendo en 3.
  Mostrar el promedio

- **Fórmula**

  Establezca a,b,c igual a 3 valores
  Establezca sum=a+b+c
  Estableca promedio=sum/3
  Imprima promedio

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

- **Diagrama de flujo**



1.3 Algoritmos

- El algoritmo antes mencionado se almacena en la memoria del computador y se ejecuta instrucción a instrucción, un ejemplo de esto seria:

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
⋮	⋮

- Luego de todos los pasos de ejecución el estado de la memoria será:

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
⋮	⋮

1.4 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.

1.6 Hardware de computación y Conceptos de almacenamiento

- Toda computadora sin importar su tamaño o costo debe realizar un conjunto mínimo de funciones y brindar capacidades para:

- Aceptar entradas
- Mostrar salidas
- Almacenar información en un formato lógico consistente
- Ejecutar operaciones aritméticas y lógicas sobre datos
- Supervisar, controlar y dirigir la operación y secuencia general del sistema

- El hardware de la computadora son los componentes que le permiten brindar estas funcionalidades

- 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 unidad de memoria almacena la información en un formato lógico consistente. Esta almacena instrucciones y datos.
- La unidad de entrada salida I/O proporciona la interfaz a la que se conectan los periféricos ej: teclados, monitores, impresoras, lectores de tarjetas, etc.

- Almacenamiento secundario:

- La **memoria principal de la computadora es volátil**. Esto implica que cuando se quita la energía se pierden 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 discos duros, cd, dvds, pendrives, etc.

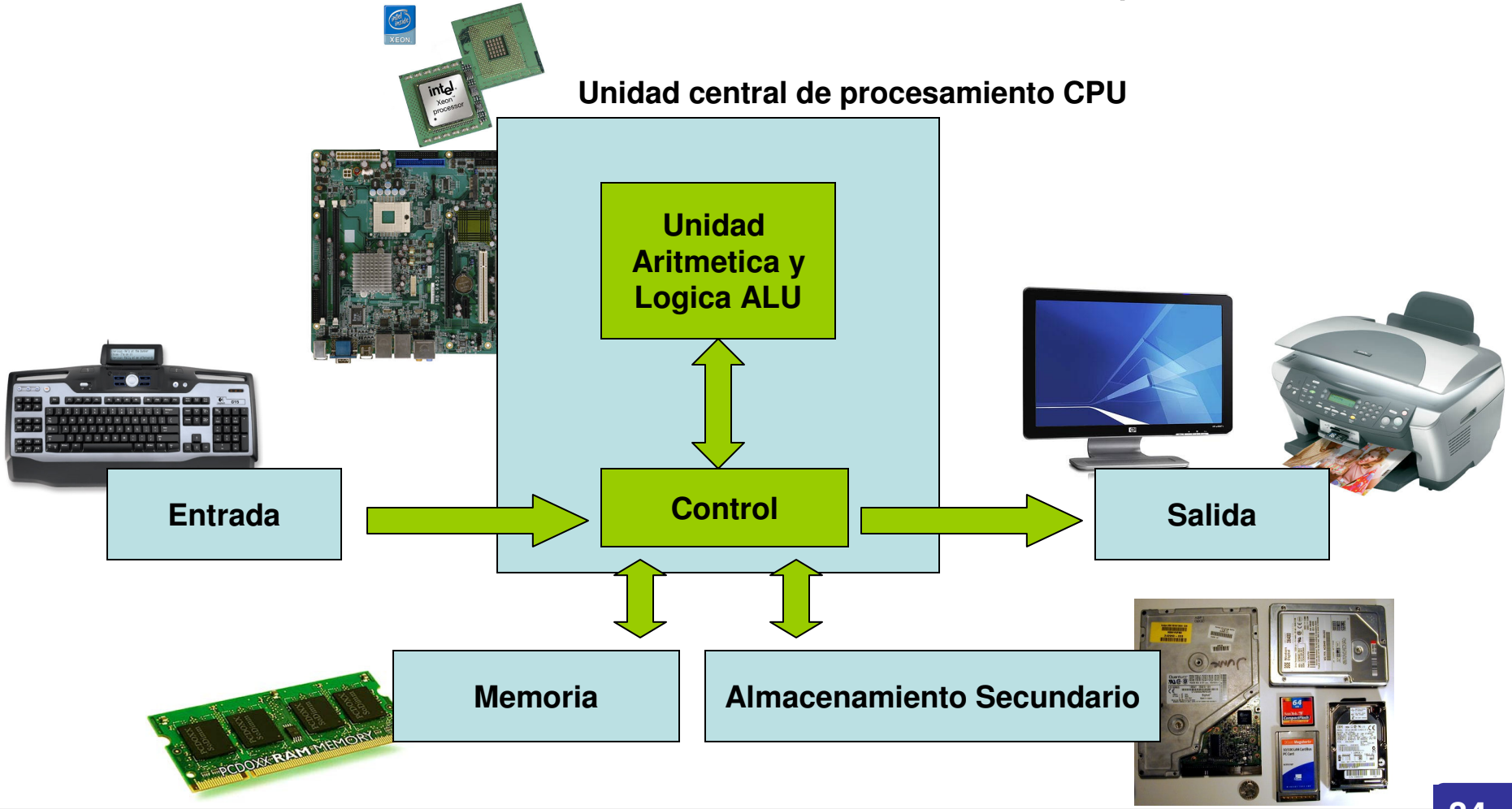
- Las primeras computadoras en los años 50 eran grandes por los relés, tubos catódicos y por almacenamiento en tarjetas perforadas. Luego en los 60 con los transistores se combino: la ALU, con la unidad de control dando lugar a la unidad central de procesamiento CPU. Esto proporciono una velocidad de procesamiento mejorada.

- Posteriormente nace el IC, circuito integrado, 100 transistores en 1 cm2.

- Las versiones actuales de estos IC contienen entre 0.7 y 1,17 billones de transistores Core i7 980x 6 núcleos a 3,33 Ghz y se conocen como chips integrados a gran escala.

1.6 Hardware de computación y Conceptos de almacenamiento

- Unidades básicas de hardware de una computadora:



- Como se almacena la información en la memoria de un computador?

- Se almacenan en secuencias de unos y ceros, porque?...

- Los computadores disponen de solo dos signos básicos, {0, 1}; pero pueden combinarse en secuencia.

- La menor variable posible de 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".**

- 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: $\sum_{i=0}^{7} b_i \cdot 2^i$

- Veamos un ejemplo: la secuencia de bits 00001011

00001011

codifica el valor

$\sum_{i=0}^{7} b_i \cdot 2^i$

$$0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 =$$

$$8 + 2 + 1 = 11$$

•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 …

- 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

- Sumaremos los valores 11 y 3 según su representación:

$$\begin{array}{rr} & 00001011 \\ + & 00000011 \\ \hline \end{array}$$

comenzando por los bits menos significativos:

$$\begin{array}{rr} \text{Acarreo} & 1 \\ & 00001011 \\ + & 00000011 \\ \hline & 0 \end{array}$$

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

$$\begin{array}{rr} \text{Acarreo} & 1\ 1 \\ & 00001011 \\ + & 00000011 \\ \hline & 10 \end{array}$$

De la forma antes mencionada llegamos al resultado:

```
Acarreo        1 1
              00001011
          +   00000011
              --------
              00001110
```

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”.

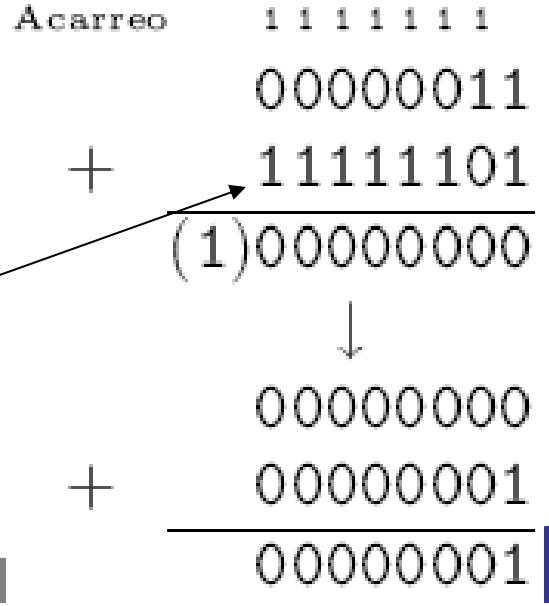
- **Complemento a uno**, para representación de números negativos. Se toma la representación posicional del numero (debe poder expresarse con 7 bits) y se invierte todos sus bits.

- 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…

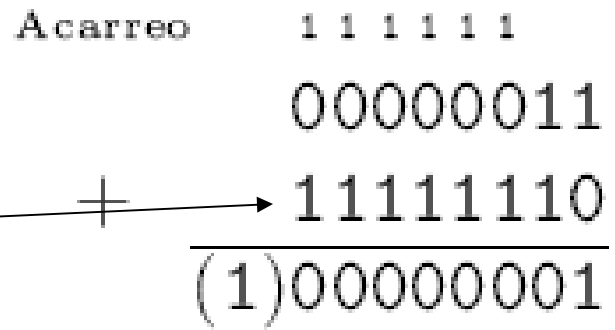


- **Complemento a dos**, utilizada por los ordenadores, mas efectiva, aunque es poco natural.

- 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.



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

1.6 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

- 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	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040		Space	64	40	100	@	@	96	60	140	`	`
33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
48	30	060	0	0	80	50	120	P	P	112	70	160	p	p