SEPARATA: CONCEPTOS BÁSICOS DE LA COMPUTACIÓN

Universidad: [Tu Universidad]
Curso: Conceptos Básicos de la Computación
Duración: 4 semanas
Elaborado para examen universitario

UNIDAD 1: HISTORIA DE LA COMPUTACIÓN

1.1 Desarrollo de la Computación a lo Largo del Tiempo

Etapas Prehistóricas de la Computación

- **Ábaco (3000 a.C.):** Primer dispositivo de cálculo conocido, utilizado en civilizaciones antiguas
- **Pascalina (1642):** Calculadora mecánica inventada por Blaise Pascal
- **Máquina de Leibniz (1694):** Primera calculadora que podía realizar las cuatro operaciones básicas

Generaciones de Computadoras

Primera Generación (1940-1956):

- Uso de tubos de vacío
- Programación en lenguaje máquina
- Ejemplos: ENIAC, UNIVAC I
- Características: Grandes, costosas, consumían mucha energía

Segunda Generación (1956-1963):

- Transistores reemplazan a los tubos de vacío
- Lenguajes de programación de alto nivel (COBOL, FORTRAN)
- Sistemas operativos primitivos
- Menor tamaño y consumo energético

Tercera Generación (1964-1971):

- Circuitos integrados
- Multiprogramación y tiempo compartido
- Minicomputadoras

- IBM System/360

Cuarta Generación (1971-presente):

- Microprocesadores
- Computadoras personales
- Redes de computadoras
- Internet y World Wide Web

Quinta Generación (1980-presente):

- Inteligencia artificial
- Procesamiento paralelo
- Reconocimiento de voz e imagen
- Computación cuántica (emergente)

1.2 Hitos Importantes en la Historia de la Tecnología

Personajes Clave

- **Charles Babbage (1791-1871):** Diseñó la Máquina Analítica, precursora de las computadoras modernas
- **Ada Lovelace (1815-1852):** Primera programadora de la historia
- **Alan Turing (1912-1954):** Padre de la ciencia de la computación, creador de la Máquina de Turing
- **John von Neumann (1903-1957):** Arquitectura de von Neumann para computadoras
- **Gordon Moore:** Ley de Moore sobre la evolución de los procesadores

Eventos Significativos

- **1945:** Arquitectura de von Neumann
- **1969:** ARPANET (precursora de Internet)
- **1971:** Primer microprocesador (Intel 4004)
- **1981:** IBM PC
- **1991:** World Wide Web se hace pública
- **2007:** Introducción del smartphone (iPhone)

UNIDAD 2: CONCEPTOS BÁSICOS DE HARDWARE

2.1 Representación de Datos: Bits y Bytes

Sistema Binario

- **Bit:** Unidad básica de información (0 o 1)
- **Byte:** Conjunto de 8 bits
- **Conversiones:**
  - 1 Byte = 8 bits
  - 1 Kilobyte (KB) = 1,024 bytes
  - 1 Megabyte (MB) = 1,024 KB
  - 1 Gigabyte (GB) = 1,024 MB
  - 1 Terabyte (TB) = 1,024 GB

Representación de Información

- **Números:** Sistema binario, octal, hexadecimal
- **Texto:** Códigos ASCII, Unicode
- **Imágenes:** Pixels, resolución, profundidad de color
- **Audio:** Muestreo digital, frecuencia, bits por muestra

2.2 Codificación de Datos

Sistemas de Numeración

- **Binario (base 2):** 0, 1
- **Octal (base 8):** 0-7
- **Decimal (base 10):** 0-9
- **Hexadecimal (base 16):** 0-9, A-F

Códigos de Caracteres

- **ASCII:** 7 bits, 128 caracteres
- **ASCII Extendido:** 8 bits, 256 caracteres
- **Unicode:** Estándar internacional para representar texto
- **UTF-8, UTF-16:** Codificaciones de Unicode

2.3 Componentes de una Computadora

Unidad Central de Procesamiento (CPU)

- **Funciones:** Ejecutar instrucciones, realizar cálculos
- **Componentes:**
  - Unidad de Control (UC)
  - Unidad Aritmético-Lógica (ALU)
  - Registros
  - Caché
- **Características:** Velocidad de reloj, núcleos, arquitectura

Memoria

- **Memoria Principal (RAM):**
  - Volátil, acceso aleatorio
  - Tipos: DRAM, SRAM, DDR
- **Memoria ROM:**
  - No volátil, solo lectura
  - Contiene firmware y BIOS
- **Jerarquía de memoria:** Registros → Caché → RAM → Almacenamiento

Almacenamiento

- **Discos Duros (HDD):**
  - Magnéticos, mecánicos
  - Gran capacidad, menor velocidad
- **Unidades de Estado Sólido (SSD):**
  - Flash memory, sin partes móviles
  - Mayor velocidad, menor capacidad
- **Almacenamiento óptico:** CD, DVD, Blu-ray

2.4 Periféricos y Dispositivos de Entrada/Salida

Dispositivos de Entrada

- **Teclado:** Entrada de texto y comandos
- **Ratón:** Dispositivo apuntador
- **Micrófono:** Entrada de audio

- **Cámara:** Captura de imagen y video
- **Escáner:** Digitalización de documentos

Dispositivos de Salida

- **Monitor:** Visualización de información
- **Impresora:** Salida en papel
- **Altavoces:** Salida de audio
- **Proyector:** Presentaciones

Dispositivos de Entrada/Salida

- **Pantallas táctiles**
- **Dispositivos de almacenamiento USB**
- **Módems y tarjetas de red**

2.5 Microprocesadores

Arquitectura del Microprocesador

- **Ciclo de instrucción:** Fetch → Decode → Execute
- **Pipeline:** Procesamiento en paralelo de instrucciones
- **Superescalaridad:** Múltiples unidades de ejecución
- **Predicción de saltos:** Optimización del flujo de instrucciones

Familias de Procesadores

- **Intel:** x86, x64, Core i3/i5/i7/i9
- **AMD:** Ryzen, EPYC
- **ARM:** Procesadores móviles y embebidos
- **Apple:** Chips M1, M2 (basados en ARM)

2.6 Arquitecturas Modernas

Arquitectura de von Neumann

- Programa y datos en la misma memoria
- Ejecución secuencial de instrucciones
- Limitaciones: Cuello de botella de von Neumann

Arquitectura Harvard

- Memorias separadas para programa y datos
- Acceso simultáneo a instrucciones y datos
- Mayor eficiencia en ciertos casos

Arquitecturas Paralelas

- **SIMD:** Una instrucción, múltiples datos
- **MIMD:** Múltiples instrucciones, múltiples datos
- **Multinúcleo:** Varios procesadores en un chip
- **GPU:** Procesamiento paralelo masivo

UNIDAD 3: CONCEPTOS BÁSICOS DE SISTEMA OPERATIVO

3.1 Funciones y Roles del Sistema Operativo

Definición

El sistema operativo es un software que actúa como intermediario entre el usuario y el hardware de la computadora, proporcionando un entorno para que otros programas puedan ejecutarse.

Funciones Principales

- **Gestión de procesos:** Creación, terminación, sincronización
- **Gestión de memoria:** Asignación y liberación de memoria
- **Gestión de archivos:** Organización y acceso a datos
- **Gestión de E/S:** Comunicación con dispositivos periféricos
- **Interfaz de usuario:** CLI o GUI
- **Seguridad:** Control de acceso y protección

3.2 Componentes de un Sistema Operativo

Núcleo (Kernel)

- **Funciones:** Gestión de recursos de bajo nivel
- **Tipos:**
  - Monolítico: Todas las funciones en espacio del kernel
  - Microkernel: Funciones mínimas en el kernel

- Híbrido: Combinación de ambos enfoques

Shell (Intérprete de comandos)

- Interfaz entre usuario y kernel
- Tipos: Línea de comandos (CLI), interfaz gráfica (GUI)

Controladores de Dispositivos (Drivers)

- Software que permite comunicación con hardware específico
- Traducen comandos del SO a instrucciones del dispositivo

3.3 Tipos de Sistemas Operativos

Por Tiempo de Respuesta

- **Tiempo Real:** Respuesta garantizada en tiempo específico
- **Tiempo Compartido:** Multiple usuarios simultáneos
- **Por Lotes:** Procesamiento de trabajos en secuencia

Por Número de Usuarios

- **Monousuario:** Un usuario a la vez (MS-DOS, Windows 9x)
- **Multiusuario:** Múltiples usuarios simultáneos (Linux, Unix)

Por Número de Tareas

- **Monotarea:** Una tarea a la vez
- **Multitarea:** Múltiples tareas aparentemente simultáneas

Por Plataforma

- **Escritorio:** Windows, macOS, Linux
- **Servidor:** Windows Server, Linux Server, Unix
- **Móvil:** Android, iOS
- **Embebido:** Sistemas específicos para dispositivos

3.4 Sistemas Operativos Específicos

Windows

- **Historia:** MS-DOS → Windows 95/98/ME → NT → XP → Vista → 7 → 8 → 10 → 11

- **Características:** GUI intuitiva, amplia compatibilidad de software
- **Arquitectura:** Híbrida, basada en NT

macOS

- **Base:** Unix (Darwin)
- **Características:** Integración con ecosistema Apple, diseño elegante
- **Versiones:** OS X → macOS (Big Sur, Monterey, Ventura, etc.)

Linux

- **Características:** Código abierto, estable, seguro
- **Distribuciones:** Ubuntu, Fedora, Debian, CentOS, RHEL
- **Uso:** Servidores, desarrollo, sistemas embebidos

3.5 Tipo de Sistemas Operativos

Centralizados

- Procesamiento en una sola máquina
- Control centralizado de recursos
- Ejemplos: Mainframes tradicionales

Distribuidos

- Procesamiento distribuido en múltiples máquinas
- Transparencia de ubicación
- Ejemplos: Sistemas en la nube, clusters

3.6 Gestión del Sistema Operativo

Gestión de Procesos

- **Proceso:** Programa en ejecución
- **Estados:** Nuevo, listo, ejecutando, bloqueado, terminado
- **Planificación:** Algoritmos para asignar CPU
- **Sincronización:** Semáforos, mutex, monitores

Gestión de Memoria

- **Memoria Virtual:** Expansión lógica de la memoria física

- **Paginación:** División de memoria en páginas
- **Segmentación:** División lógica del espacio de direcciones
- **Algoritmos de reemplazo:** FIFO, LRU, Optimal

Gestión de Periféricos

- **Controladores de dispositivos**
- **Interrupciones:** Señales de hardware/software
- **DMA:** Acceso directo a memoria
- **Spooling:** Cola de trabajos de impresión

Gestión de Archivos

- **Sistema de archivos:** Organización de datos en almacenamiento
- **Tipos:** FAT32, NTFS, ext4, APFS, ZFS
- **Operaciones:** Crear, leer, escribir, eliminar
- **Directorios:** Estructura jerárquica de organización

UNIDAD 4: INTRODUCCIÓN A LA TEORÍA DE COMPILADORES

4.1 Análisis Léxico

Definición

Primera fase de la compilación que convierte la secuencia de caracteres del código fuente en una secuencia de tokens (símbolos léxicos).

Componentes

- **Scanner/Lexer:** Programa que realiza el análisis léxico
- **Tokens:** Unidades léxicas básicas (palabras reservadas, identificadores, operadores)
- **Lexemas:** Secuencia de caracteres que forman un token
- **Patrones:** Reglas que describen los lexemas de un token

Proceso

1. Lee caracteres del código fuente
2. Agrupa caracteres en lexemas
3. Identifica el token correspondiente

4. Pasa tokens al analizador sintáctico

Herramientas

- **Lex/Flex:** Generadores de analizadores léxicos
- **Expresiones regulares:** Definen patrones de tokens

4.2 Análisis Sintáctico

Definición

Segunda fase que verifica que los tokens formen construcciones sintácticas válidas según la gramática del lenguaje.

Conceptos Fundamentales

- **Gramática:** Conjunto de reglas que definen la sintaxis
- **Gramáticas libres de contexto:** Tipo de gramática usado en lenguajes de programación
- **Árbol de análisis:** Representación gráfica de la derivación
- **Árbol sintáctico abstracto (AST):** Versión simplificada del árbol de análisis

Tipos de Analizadores

- **Descendentes (Top-Down):**
  - Recursivo descendente
  - LL(k) - Left-to-right, Leftmost, k symbols lookahead
- **Ascendentes (Bottom-Up):**
  - LR(k) - Left-to-right, Rightmost, k symbols lookahead
  - LALR (Look-Ahead LR)
  - SLR (Simple LR)

4.3 Análisis Sintáctico Descendente

Características

- Construye el árbol de análisis desde la raíz hacia las hojas
- Utiliza derivaciones por la izquierda
- Más intuitivo para entender

Recursivo Descendente

- Cada no terminal de la gramática corresponde a una función
- Fácil de implementar manualmente
- Puede tener problemas con recursión izquierda

Analizadores LL(k)

- **LL(1):** Más común, usa un símbolo de anticipación
- **Tabla de análisis:** Guía las decisiones de parsing
- **Eliminación de recursión izquierda:** Transformación necesaria
- **Factorización izquierda:** Elimina ambigüedad

4.4 Análisis Sintáctico Ascendente

Características

- Construye el árbol desde las hojas hacia la raíz
- Utiliza derivaciones por la derecha en reversa
- Más potente que los descendentes

Análisis Shift-Reduce

- **Shift:** Mueve un token de entrada a la pila
- **Reduce:** Reemplaza símbolos en la pila por un no terminal
- **Estado:** Configuración actual del parser

Analizadores LR

- **LR(0):** Sin anticipación
- **SLR:** Simple LR con follow sets
- **LR(1):** Un símbolo de anticipación
- **LALR:** Combinación de estados LR(1)

4.5 Tabla de Tipos y de Símbolos

Tabla de Símbolos

- **Propósito:** Almacenar información sobre identificadores
- **Información guardada:**
  - Nombre del identificador
  - Tipo de dato

- Ámbito (scope)
- Dirección de memoria
- Atributos adicionales

Gestión de Ámbitos

- **Ámbito global:** Visible en todo el programa
- **Ámbito local:** Visible solo en un bloque
- **Anidamiento:** Ámbitos dentro de otros ámbitos
- **Reglas de visibilidad:** Cómo se resuelven los nombres

Tabla de Tipos

- **Tipos básicos:** int, float, char, boolean
- **Tipos compuestos:** arrays, estructuras, uniones
- **Verificación de tipos:** Compatibilidad en operaciones
- **Coerción:** Conversión automática de tipos

4.6 Generación de Código

Fases de Generación

1. **Generación de código intermedio**
2. **Optimización de código**
3. **Generación de código objeto**

Código Intermedio

- **Representaciones:**
  - Código de tres direcciones
  - Árbol sintáctico abstracto
  - Notación postfija
- **Ventajas:** Independiente de la máquina, facilita optimización

Optimización

- **Tipos:**
  - Optimización de mirilla (peephole)
  - Optimización local (bloque básico)

    - Optimización global (flujo de datos)
- **Técnicas:**
    - Eliminación de código muerto
    - Propagación de constantes
    - Eliminación de subexpresiones comunes

4.7 Lenguajes de Programación

Paradigmas

- **Imperativo:** C, Pascal, COBOL
- **Orientado a objetos:** Java, C++, Python
- **Funcional:** Haskell, Lisp, ML
- **Lógico:** Prolog
- **Híbrido:** JavaScript, Scala, F#

Características de los Lenguajes

- **Sintaxis:** Forma de escribir construcciones
- **Semántica:** Significado de las construcciones
- **Pragmática:** Uso práctico del lenguaje

4.8 Tipos de Lenguajes de Programación

Por Nivel de Abstracción

- **Bajo nivel:**
    - Lenguaje máquina: Código binario
    - Ensamblador: Mnemonicos para instrucciones
- **Alto nivel:**
    - Más cercanos al lenguaje natural
    - Portables entre diferentes arquitecturas
    - Ejemplos: C, Java, Python, JavaScript

Por Generación

- **Primera generación:** Lenguaje máquina
- **Segunda generación:** Lenguaje ensamblador

- **Tercera generación:** Lenguajes de alto nivel (C, COBOL)
- **Cuarta generación:** Lenguajes de consulta (SQL)
- **Quinta generación:** Lenguajes de inteligencia artificial (Prolog)

Por Paradigma de Programación

- **Estructurado:** Secuencia, selección, iteración
- **Modular:** División en módulos o funciones
- **Orientado a objetos:** Encapsulación, herencia, polimorfismo
- **Funcional:** Funciones como elementos de primera clase
- **Declarativo:** Describe qué se quiere, no cómo hacerlo

BIBLIOGRAFÍA Y FUENTES RECOMENDADAS

Libros de Texto Principales

Historia de la Computación:

1. Ceruzzi, Paul E. "A History of Modern Computing" (2nd Edition)
2. Campbell-Kelly, Martin. "Computer: A History of the Information Machine"
3. Ifrah, Georges. "The Universal History of Computing"

Hardware y Arquitectura de Computadores:

1. Patterson, David A. & Hennessy, John L. "Computer Organization and Design: The Hardware/Software Interface" (5th Edition)
2. Stallings, William. "Computer Organization and Architecture" (10th Edition)
3. Tanenbaum, Andrew S. "Structured Computer Organization" (6th Edition)

Sistemas Operativos:

1. Silberschatz, Abraham; Galvin, Peter; Gagne, Greg. "Operating System Concepts" (10th Edition)
2. Tanenbaum, Andrew S. "Modern Operating Systems" (4th Edition)
3. Stallings, William. "Operating Systems: Internals and Design Principles" (8th Edition)

Compiladores y Lenguajes de Programación:

1. Aho, Alfred V.; Lam, Monica S.; Sethi, Ravi; Ullman, Jeffrey D. "Compilers: Principles, Techniques, and Tools" (2nd Edition) - "Dragon Book"

2. Appel, Andrew W. "Modern Compiler Implementation in Java/C/ML"
3. Cooper, Keith D.; Torczon, Linda. "Engineering a Compiler" (2nd Edition)

Recursos en Línea

Sitios Web Académicos:

- MIT OpenCourseWare (Introducción a la Ciencia de la Computación)
- Stanford CS Education Library
- Carnegie Mellon Computer Science Department
- Berkeley CS Division Course Materials

Documentación Técnica:

- Intel Developer Zone
- AMD Developer Guides
- Microsoft Developer Network (MSDN)
- Linux Documentation Project

Tutoriales y Cursos:

- Coursera: Computer Science Courses
- edX: Introduction to Computer Science
- Khan Academy: Intro to Programming
- Codecademy: Computer Science Path

Artículos y Papers Relevantes

1. Von Neumann, John. "First Draft of a Report on the EDVAC" (1945)
2. Moore, Gordon E. "Cramming More Components onto Integrated Circuits" (1965)
3. Turing, Alan M. "On Computable Numbers" (1936)
4. Dijkstra, Edsger W. "The Structure of the 'THE'-Multiprogramming System" (1968)

Revistas Especializadas

- Communications of the ACM
- IEEE Computer
- ACM Computing Surveys
- IEEE Software

- Computer Architecture Letters

EJERCICIOS DE PRÁCTICA PARA EXAMEN

Historia de la Computación

1. Elabora una línea de tiempo con los principales hitos en la historia de las computadoras
2. Compara las características de cada generación de computadoras
3. Explica la importancia de la Ley de Moore y su impacto actual

Hardware

1. Convierte números entre diferentes bases (binario, octal, decimal, hexadecimal)
2. Calcula capacidades de almacenamiento en diferentes unidades
3. Describe el ciclo de instrucción de un procesador
4. Diseña la configuración de hardware para diferentes tipos de usuarios

Sistemas Operativos

1. Compara ventajas y desventajas de diferentes sistemas operativos
2. Describe los estados de un proceso y las transiciones entre ellos
3. Explica algoritmos de planificación de procesos
4. Analiza diferentes sistemas de archivos y sus características

Compiladores

1. Diseña un analizador léxico simple para un lenguaje básico
2. Construye árboles sintácticos para expresiones matemáticas
3. Implementa una tabla de símbolos básica
4. Compara diferentes paradigmas de programación con ejemplos

---

Nota: Esta separata ha sido elaborada como material de estudio para examen universitario. Se recomienda complementar con las fuentes bibliográficas mencionadas y la participación activa en clases teóricas y prácticas.