MANUAL DE PRÁCTICAS DE LABORATORIO

Introducción a la Tecnología de Compiladores

Programa Académico	**Ing. en Software**
Plan de Estudios	**21**
Fecha de elaboración	**06/11/2025**
Versión del Documento	**1.0**

UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

Dra. Martha Patricia Patiño Fierro
Rectora

Mtra. Ana Lisette Valenzuela Molina
Encargada del Despacho de la Secretaría General Académica

Mtro. José Antonio Romero Montaño
Secretario General Administrativo

Lic. Jorge Omar Herrera Gutiérrez
Encargado de Despacho de Secretario General de Planeación

Tabla de contenido

INTRODUCCIÓN

Como parte de las herramientas esenciales para la formación académica de los estudiantes de la Universidad Estatal de Sonora, se definen manuales de práctica de laboratorio como elemento en el cual se define la estructura normativa de cada práctica y/o laboratorio, además de representar una guía para la aplicación práctica del conocimiento y el desarrollo de las competencias clave en su área de estudio. Su diseño se encuentra alineado con el modelo educativo institucional, el cual privilegia el aprendizaje basado en competencias, el aprendizaje activo y la conexión con escenarios reales.

Con el propósito de fortalecer la autonomía de los estudiantes, su pensamiento crítico y sus habilidades para la resolución de problemas, las prácticas de laboratorio integran estrategias didácticas como el aprendizaje basado en proyectos, el trabajo colaborativo, la experimentación guiada y el uso de tecnologías educativas. De esta manera, se promueve un proceso de enseñanza-aprendizaje dinámico, en el que los estudiantes no solo adquieren conocimientos teóricos, sino que también desarrollan habilidades prácticas y reflexivas para su desempeño profesional.

- Propósito del manual

El propósito del manual de laboratorio es proporcionar una guía estructurada para el desarrollo de prácticas relacionadas con Introducción a la Tecnología de Compiladores. Estas prácticas buscan fortalecer la comprensión teórica y técnica de los estudiantes, fomentar el pensamiento lógico y computacional, iimplementando compiladores básicos ó desarrollar estos sistemas, mediante el análisis de escenarios donde no existan recursos específicos, alineados al perfil de egreso del programa académico.

- Justificación de su uso en el programa académico

  El manual se justifica como herramienta didáctica esencial que facilita el desarrollo de competencias profesionales, disciplinares y blandas requeridas por los estudiantes del programa de Ingeniería en Software. A través de experiencias prácticas, se promueve el aprendizaje activo, significativo y por competencias, en concordancia con el enfoque educativo de la UES. Permite además evaluar el desempeño de manera objetiva mediante rúbricas institucionales, garantizando la calidad y pertinencia del proceso de enseñanza-aprendizaje.

- Competencias a desarrollar
  - **Competencias blandas:**
    Análisis
    Organización

Responsabilidad
innovación
Ética profesional en el desarrollo de software.
Trabajo colaborativo en el desarrollo de soluciones tecnológicas.
Uso eficiente de tecnologías de información y herramientas de programación.
Habilidades de comunicación escrita (reportes técnicos).

- **Competencias disciplinares:**

  Identificar los conceptos básicos sobre compiladores, intérpretes y metalenguajes, comprendiendo sus características y estructura.
  Comprender las características de los distintos tipos de traductores y compiladores, a fin de entender sus funciones y la importancia que tienen.
  Ejercer la capacidad de análisis y solución de problemas en la conversión y manejo de autómatas finitos, base para el análisis léxico.
  Realizar el análisis y aplicación del diseño de un analizador sintáctico, especificando la gramática y utilizando el diseño descendente
  Construir y utilizar una tabla de símbolos para almacenar información semántica relevante y aplicar reglas semánticas básicas durante el análisis.
  Construir un flujo completo de análisis léxico-sintáctico-semántico con generación de código intermedio integrando diferentes herramientas (jflex + CUP/Bison).
  Aplicar soluciones e innovaciones tecnológicas con la finalidad de automatizar los procesos.

- **Competencias profesionales:**

  Diseño e implementación Compiladores básicos
  Aplicación de técnicas de programación para el procesamiento de datos y automatización de procesos.
  Desarrollo de proyectos tecnológicos integradores con enfoque de innovación.
  Resolución de problemas reales mediante el uso de tecnologías emergentes.

IDENTIFICACIÓN

Nombre de la Asignatura		Introducción a la Tecnología de Compiladores	
Clave	061CP027	**Créditos**	**6**
Asignaturas Antecedentes		**Plan de Estudios**	**2021**

Área de Competencia	Competencia del curso
Emplear el pensamiento estratégico en la gestión empresarial, a nivel regional, nacional o internacional, mediante la aplicación efectiva de herramientas metodológicas, de producción, financieras, mercadológicas y de gestión del capital humano, con el fin de incrementar los índices de productividad y competitividad organizacional, bajo un enfoque de calidad, análisis de problemas, trabajo en equipo y toma de decisiones.	Implementar compiladores básicos ó desarrollar estos sistemas, mediante el análisis de escenarios donde no existan recursos específicos para un propósito particular, aprovechando eficazmente las técnicas y tecnologías de programación para la solución de problemas dentro de las organizaciones, con un alto sentido de responsabilidad y trabajo en equipo.

Carga Horaria de la asignatura

Horas Supervisadas			Horas Independientes	Total de Horas
Aula	Laboratorio	Plataforma		
3	1	1	2	7

Consignación del Documento

Unidad Académica	Unidad Académica Navojoa
Fecha de elaboración	06/11/2025
Responsables del diseño	Filiberto Valenzuela Mendoza
Validación	
Recepción	Coordinación de Procesos Educativos

NORMAS DE SEGURIDAD Y BUENAS PRÁCTICAS

Reglamento general del laboratorio

Acceso a Laboratorio de Programación

- El acceso a las salas de trabajo será sólo a las horas que no haya clases.
- Deberá entrar sólo con sus documentos de trabajo, el resto de sus cosas deberá dejarlo a la entrada del laboratorio de cómputo.
- No deberá entrar a las salas de trabajo con alimentos, bebidas o fumando.
- Queda estrictamente prohibido introducir animales.

Permanencia

- Antes de trabajar se debe revisar el equipo de cómputo donde fue designado y reportar alguna anomalía al encargado de la sala; si posteriormente se detecta alguna falla no reportada, se le responsabilizará sobre ésta.
- No deberá mover el equipo de cómputo ni mobiliario de su lugar bajo ningún motivo; verifique en su apartado el equipo que va a necesitar para su trabajo.
- El usuario que dañe el equipo de cómputo, sus instalaciones o cambie su configuración del software, incluso protectores de pantalla o colores de ventanas quedará responsabilizado de pagar su costo de reparación o adquisición.
- El comportamiento de todo usuario no debe ir en contra de la moral y las buenas costumbres dentro de las diferentes salas de cómputo.
- El usuario deberá mantener limpia su área de trabajo.
- El uso del equipo de cómputo es exclusivamente para fines didácticos y de investigación, por lo que se prohíbe el uso de juegos, trabajos personales ó de terceros con fines de lucro.
- Solo se permitirá el trabajo individual por computadora, a excepciones de aquellos cursos ó clases impartidas por instructores ó profesores.
- Los usuarios no podrán ingresar ningún tipo de radio, grabadora ó elementos que produzcan ruido y/o elementos magnéticos.
- Los estudiantes deben guardar respeto mantener los canales de comunicación y demás normas establecidas con los monitores y responsables del centro de cómputo y estos para con ellos.
- El acceso a los laboratorios es para el usuario que haya solicitado el servicio, bajo ninguna circunstancia podrán estar dos personas trabajando conjuntamente en una computadora.

Uso adecuado del equipo y materiales

- El equipo periférico que se presta, es solo para uso durante la sesión de trabajo por lo que deberá de devolverlo al término de esta.
- Para asesorías ó dudas con respecto al uso del equipo, debe dirigirse al responsable en turno dentro de la sala donde se encuentre ó bien dirigirse al personal encargado del laboratorio
- El usuario no debe desconectar ni destapar los ratones de las computadoras, las impresoras ó cualquier equipo periférico que se tenga instalado.
- Es obligación del usuario saber operar el equipo de cómputo y periféricos; debe consultar con el personal del servicio social o leer los folletos sobre el uso de periféricos distribuidos para tal efecto; ya que el daño por el mal uso será responsabilidad del usuario.
- En caso de necesitar tóner la impresora notifique al responsable en turno ó bien al encargado del laboratorio, para que el la verifique y éste sea remplazado.

Procedimientos en caso de emergencia

- Evacuación ordenada por rutas señalizadas.
- Uso de extintores solo por personal capacitado.
- Reporte inmediato de accidentes al personal responsable.

MATRIZ DE CORRESPONDENCIA

Señalar la relación de cada práctica con las competencias del perfil de egreso

PRÁCTICA	PERFIL DE EGRESO
Práctica No. 1: Análisis Histórico y Conceptual de Traductores (Investigación)	Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo.
Práctica No. 2: Clasificación de Traductores y Compiladores (Cuadro Sinóptico)	Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo.
Práctica No. 3: Solución de Ejercicios de Autómatas Finitos (AFD y AFND)	Aplicar soluciones e innovaciones tecnológicas con la finalidad de automatizar los procesos, atendiendo los principios de la organización y gestión efectiva de la información.
Práctica No. 4: Construcción de un Analizador Léxico con Herramientas Computacionales	Aplicar soluciones e innovaciones tecnológicas con la finalidad de automatizar los procesos, atendiendo los principios de la organización y gestión efectiva de la información.
Práctica No. 5: Diseño e Implementación de un Analizador Sintáctico Descendente (DAS)	Aplicar soluciones e innovaciones tecnológicas con la finalidad de automatizar los procesos, atendiendo los principios de la organización y gestión efectiva de la información.
Práctica No. 6: Desarrollo de Analizador Semántico con Tabla de Símbolos	Aplicar soluciones e innovaciones tecnológicas con la finalidad de automatizar los procesos, atendiendo los principios de la organización y gestión efectiva de la información.
Práctica No. 7: Diseño de Gramática con ANTLR4 para Generación de Código Intermedio	Aplicar soluciones e innovaciones tecnológicas con la finalidad de automatizar los procesos, atendiendo los principios de la organización y gestión efectiva de la información.
Práctica No. 8: Integración de Herramientas (jflex + CUP/Bison) para un Compilador Completo	Aplicar soluciones e innovaciones tecnológicas con la finalidad de automatizar los procesos, atendiendo los principios de la organización y gestión efectiva de la información.

RELACIÓN DE PRÁCTICAS DE LABORATORIO POR ELEMENTO DE COMPETENCIA

Elemento de Competencia al que pertenece la práctica	**EC I** Identificar los conceptos básicos sobre compiladores, intérpretes y metalenguajes, con el fin de comprender sus características y estructura según los estándares establecidos por las metodologías de desarrollo, aplicándolos en la solución de problemas dentro de las organizaciones, demostrando capacidad de análisis, organización y responsabilidad.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 1	Análisis Histórico y Conceptual de Traductores (Investigación)	Identificar los conceptos básicos sobre compiladores, intérpretes y metalenguajes, comprendiendo sus características y estructura.
Práctica No. 2	Clasificación de Traductores y Compiladores (Cuadro Sinóptico)	Comprender las características de los distintos tipos de traductores y compiladores, a fin de entender sus funciones y la importancia que tienen.

Elemento de Competencia al que pertenece la práctica	**EC II** Diseñar analizadores léxicos utilizando las técnicas de desarrollo de autómatas finitos y de expresiones regulares, empleados en el desarrollo de gramáticas libres de contexto, siguiendo los estándares de calidad de software establecidos y aplicándolos en la solución de problemas dentro de las organizaciones, demostrando capacidad de análisis y trabajo en equipo.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 3	Solución de Ejercicios de Autómatas Finitos (AFD y AFND)	Ejercer la capacidad de análisis y solución de problemas en la conversión y manejo de autómatas finitos, base para el análisis léxico.
Práctica No. 4	Construcción de un Analizador Léxico con Herramientas Computacionales	Solucionar ejercicios de construcción de un analizador léxico mediante el uso de herramientas computacionales, aplicando los conceptos de lenguajes y expresiones regulares.

Elemento de Competencia al que pertenece la práctica	EC III
	Diseñar un analizador sintáctico y semántico descendente, para dar solución a las tareas de traducción del código fuente para la construcción de un lenguaje de programación bajo los estándares y buenas prácticas para el desarrollo de software aplicable en las organizaciones, con un alto sentido de la innovación y trabajo en equipo.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 5	Diseño e Implementación de un Analizador Sintáctico Descendente (DAS)	Realizar el análisis y aplicación del diseño de un analizador sintáctico, especificando la gramática y utilizando el diseño descendente
Práctica No. 6	Desarrollo de Analizador Semántico con Tabla de Símbolos	Construir y utilizar una tabla de símbolos para almacenar información semántica relevante y aplicar reglas semánticas básicas durante el análisis

Elemento de Competencia al que pertenece la práctica	EC IV
	Construir un prototipo de generador de código intermedio para dar solución a una problemática especifica dentro de las organizaciones para la cual no exista una alternativa comercial, mediante el análisis y resolución de problemas, con un enfoque de calidad y buenas prácticas establecidas para el desarrollo de software.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 7	Diseño de Gramática con ANTLR4 para Generación de Código Intermedio	Diseñar una gramática e incluir acciones semánticas en ANTLR4 para la construcción de un Árbol de Sintaxis Abstracta (AST) y la generación de código de tres direcciones.
Práctica No. 8	Integración de Herramientas (jflex + CUP/Bison) para un Compilador Completo	Construir un flujo completo de análisis léxico-sintáctico-semántico con generación de código intermedio integrando diferentes herramientas (jflex + CUP/Bison).

UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

PRÁCTICAS

NOMBRE DE LA PRÁCTICA	Práctica No. 1: Análisis Histórico y Conceptual de Traductores (Investigación)
COMPETENCIA DE LA PRÁCTICA	Identificar los conceptos básicos sobre compiladores, intérpretes y metalenguajes, comprendiendo sus características y estructura.

FUNDAMENTO TÉORICO

La teoría de compiladores se basa en comprender cómo el software traduce un lenguaje de alto nivel a uno de bajo nivel. Es crucial conocer la evolución histórica (desde los años 40) y diferenciar entre conceptos clave como compilador, intérprete, traductor, lenguaje fuente/objeto, ensamblador y código máquina.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Recursos bibliográficos (libros y artículos como los citados).
Plataforma educativa institucional.
Computadora con acceso a internet y software de procesamiento de texto/PDF

PROCEDIMIENTO O METODOLOGÍA

1. Investigación Bibliográfica: Consultar las fuentes recomendadas y otras con sustento académico para construir una perspectiva histórica de la teoría de compiladores e intérpretes (de 1940 a la actualidad).
2. Definición de Conceptos: Incluir las definiciones y el rol de: compilador, intérprete, traductor, lenguaje de programación, lenguaje fuente, lenguaje objeto, ensamblador, código máquina y nemotécnicos.
3. Análisis Comparativo: Redactar un apartado que compare las ventajas y desventajas del uso de compiladores frente a intérpretes.
4. Elaboración del Documento: Estructurar el trabajo de investigación (introducción, desarrollo de conceptos, análisis comparativo, conclusión y referencias).
5. Entrega: Guardar el documento en formato PDF y subirlo a la plataforma educativa institucional.

RESULTADOS ESPERADOS

Un trabajo de investigación completo y bien fundamentado, en formato PDF, que demuestre la comprensión de los conceptos básicos y la evolución histórica de los traductores de lenguaje.

ANÁLISIS DE RESULTADOS

El análisis debe enfocarse en la claridad de las definiciones y la coherencia histórica presentada. Discutir cómo la evolución de estas herramientas impactó directamente en el desarrollo de la programación moderna.

CONCLUSIONES Y REFLEXIONES
Reflexionar sobre la importancia de estos conceptos como base para el desarrollo de software y la construcción de nuevos lenguajes de programación. Mencionar la competencia blanda promovida (organización y responsabilidad).

ACTIVIDADES COMPLEMENTARIAS
Participar en el foro de discusión "Compiladores como eje principal en el desarrollo de software", argumentando su importancia en relación con Arquitectura de Computadoras, Lenguajes de Programación, Teoría de Lenguajes, Teoría de Algoritmos e Ingeniería de Software.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Documento en formato PDF
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de Investigación de Conceptos.
Formatos de reporte de prácticas	Documento PDF de la investigación subido a plataforma.

NOMBRE DE LA PRÁCTICA	Práctica No. 2: Clasificación de Traductores y Compiladores (Cuadro Sinóptico)
COMPETENCIA DE LA PRÁCTICA	Comprender las características de los distintos tipos de traductores y compiladores, a fin de entender sus funciones y la importancia que tienen.

FUNDAMENTO TÉORICO

El conocimiento de la clasificación de traductores es vital para entender las arquitecturas y modelos de ejecución del software. Se distinguen tipos como Compiladores cruzados, JIT, optimizadores, de una/varias pasadas, Intérpretes, Preprocesadores, Ensambladores y Conversores fuente-fuente.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Información proporcionada por el facilitador en el aula.
Recursos y materiales de la sección del curso.
Fuentes con sustento académico.
Software para crear cuadros sinópticos (ej. Word, Lucidchart, Canva).

PROCEDIMIENTO O METODOLOGÍA

1. **Recolección de Información**: Identificar y recopilar los aspectos más importantes de los distintos tipos de traductores y compiladores mencionados en el fundamento teórico.
2. **Estructura Jerárquica**: Diseñar la estructura del cuadro sinóptico, usando llaves y una jerarquía clara para agrupar la información (ej. Traductores, Compiladores, Tipos de Compiladores).
3. **Elaboración del Cuadro**: Integrar las características, funciones y diferencias clave de cada tipo de compilador/traductor.
4. **Revisión y Formato**: Asegurarse de que el cuadro sinóptico sea claro y conciso, optimizando la presentación visual.
5. **Entrega**: **Guardar el documento en formato PDF** y subirlo a la plataforma educativa institucional.

RESULTADOS ESPERADOS

Un cuadro sinóptico en formato PDF que clasifique de manera clara y jerárquica los distintos tipos de traductores y compiladores, facilitando su comprensión.

ANÁLISIS DE RESULTADOS

Analizar cómo la clasificación de los compiladores (tipos y modelos de ejecución) influye en la eficiencia y el uso que se les da en diferentes contextos (ej. Compiladores JIT en entornos de ejecución dinámica).

CONCLUSIONES Y REFLEXIONES
Concluir sobre el impacto práctico de elegir un tipo de traductor sobre otro en el desarrollo de sistemas (ej. sistemas operativos vs. aplicaciones web). Mencionar la competencia blanda promovida (análisis y organización).

ACTIVIDADES COMPLEMENTARIAS
Elaborar una infografía sobre los principios de construcción de compiladores (grafos sintácticos, gramáticas, sintaxis, semántica, notación BNF, etc.).

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Cuadro Sinóptico Infografía
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de Cuadro Sinóptico.
Formatos de reporte de prácticas	Documento PDF del cuadro sinóptico subido a plataforma.

NOMBRE DE LA PRÁCTICA	Práctica No. 3: Solución de Ejercicios de Autómatas Finitos (AFD y AFND)
COMPETENCIA DE LA PRÁCTICA	Ejercer la capacidad de análisis y solución de problemas en la conversión y manejo de autómatas finitos, base para el análisis léxico.

FUNDAMENTO TÉORICO

Los autómatas finitos (AFD - Deterministas y AFND - No Deterministas) son modelos matemáticos esenciales para reconocer patrones léxicos en la primera fase de la compilación. Es crucial dominar la transformación de un AFND a un AFD.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Ejercicios de autómatas propuestos por el facilitador.
Material de apoyo del curso (ej. libros sobre Teoría de Autómatas y Lenguajes Formales) .
Laboratorio de cómputo.
Software de simulación de autómatas (opcional).

PROCEDIMIENTO O METODOLOGÍA

1. **Revisión Teórica:** Repasar los conceptos de autómata, AFD, AFND y el procedimiento de transformación de AFND a AFD.
2. **Análisis de Ejercicios:** Recibir los ejercicios propuestos por el facilitador, que deben incluir: diseño de AFND, diseño de AFD y ejercicios de conversión AFND a AFD.
3. **Solución Individual:** Resolver los ejercicios, mostrando claramente el procedimiento y las tablas de transición (o diagramas de estado) para cada autómata.
4. **Prueba y Verificación:** En el laboratorio, si es posible, usar un simulador o herramienta de verificación para comprobar que los autómatas aceptan o rechazan las cadenas de lenguaje correctas.
5. **Entrega:** Documentar las soluciones de forma clara (a mano o digital) y entregar el reporte individual en la plataforma/laboratorio.

RESULTADOS ESPERADOS

Solución individual correcta de los ejercicios propuestos, demostrando el dominio de las técnicas de diseño y transformación de autómatas finitos deterministas y no deterministas.

ANÁLISIS DE RESULTADOS

Comparar la eficiencia y facilidad de implementación de un AFND y un AFD para el mismo lenguaje.
Analizar cómo el proceso de conversión elimina la ambigüedad en el reconocimiento de tokens.

CONCLUSIONES Y REFLEXIONES

Concluir sobre la importancia de los autómatas como fundamento teórico para la construcción del analizador léxico de un compilador. Mencionar la competencia blanda promovida (análisis y resolución de problemas).

ACTIVIDADES COMPLEMENTARIAS
Elaborar un cuadro sinóptico sobre la caracterización de los autómatas programables, identificando los conceptos principales de autómata, AFD, AFND y transformación.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Ejercicios resueltos Cuadro Sinóptico
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de solución individual de ejercicios.
Formatos de reporte de prácticas	Reporte de ejercicios resueltos (digital o físico).

NOMBRE DE LA PRÁCTICA	Práctica No. 4: Construcción de un Analizador Léxico con Herramientas Computacionales
COMPETENCIA DE LA PRÁCTICA	Solucionar ejercicios de construcción de un analizador léxico mediante el uso de herramientas computacionales, aplicando los conceptos de lenguajes y expresiones regulares.

FUNDAMENTO TÉORICO

El analizador léxico (lexer) es el encargado de agrupar caracteres en tokens (unidades léxicas). Su construcción automática se basa en la definición de expresiones regulares para cada patrón léxico, utilizando herramientas como Lex/Flex, JFlex o ANTLR4.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Ejercicios de construcción de analizador léxico propuestos.
Computadoras en el laboratorio.
Herramientas de construcción automática: Lex/Flex/JFlex/ANTLR4.
Recursos sobre Lenguajes Regulares y Gramáticas

PROCEDIMIENTO O METODOLOGÍA

1. **Definición de Tokens:** Recibir un lenguaje de ejemplo sencillo (ej. calculadora básica o subconjunto de C) y definir los tokens (palabras clave, identificadores, operadores, etc.).
2. **Especificación de Patrones:** Escribir las expresiones regulares para cada token en el formato de la herramienta seleccionada (ej. JFlex o ANTLR4).
3. **Lógica del Analizador:** Configurar la herramienta para que asocie cada patrón con una acción (ej. devolver el token, registrar en tabla de símbolos simple).
4. **Gestión de Errores:** Implementar una lógica básica para el manejo de caracteres no reconocidos o errores léxicos.
5. **Pruebas:** Ejecutar el analizador léxico con cadenas de entrada válidas e inválidas para verificar que los tokens se generan correctamente y que los errores se detectan.
6. **Entrega:** Presentar el código fuente del analizador y las pruebas de ejecución en el laboratorio.

RESULTADOS ESPERADOS

Un analizador léxico funcional construido con una herramienta como JFlex o ANTLR4, capaz de identificar y clasificar los tokens de un lenguaje específico.

ANÁLISIS DE RESULTADOS

Analizar la cobertura de las expresiones regulares. Evaluar la eficiencia del analizador léxico para el reconocimiento de patrones y la detección de errores.

CONCLUSIONES Y REFLEXIONES
Concluir sobre la importancia de la automatización en la fase léxica del compilador, y cómo las expresiones regulares son una herramienta poderosa para esta tarea. Mencionar la competencia blanda promovida (solución de problemas y trabajo en equipo si se realiza en grupo).

ACTIVIDADES COMPLEMENTARIAS
Elaborar un mapa conceptual sobre lenguajes regulares y expresiones regulares.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Código fuente del analizador léxico y reporte de pruebas de ejecución. Mapa conceptual
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de solución individual de ejercicios.
Formatos de reporte de prácticas	Código fuente del analizador léxico y reporte de pruebas de ejecución.

NOMBRE DE LA PRÁCTICA	Práctica No. 5: Diseño e Implementación de un Analizador Sintáctico Descendente (DAS)
COMPETENCIA DE LA PRÁCTICA	Realizar el análisis y aplicación del diseño de un analizador sintáctico, especificando la gramática y utilizando el diseño descendente.

FUNDAMENTO TÉORICO

El análisis sintáctico verifica si la secuencia de tokens se ajusta a la gramática del lenguaje (estructura). El diseño descendente (ej. Recursive Descent Parser o herramientas como ANTLR4) construye el árbol sintáctico (AST) de arriba hacia abajo, partiendo del símbolo inicial.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Caso de estudio (lenguaje sencillo) proporcionado por el facilitador.
Computadoras en el aula/laboratorio.
Herramientas de construcción automática: Lex/Flex/JFlex/ANTLR4.
Recursos sobre Analizadores Sintácticos

PROCEDIMIENTO O METODOLOGÍA

1. **Análisis del Caso**: En **equipos de 3 personas**, analizar el lenguaje de programación sencillo del caso de estudio, identificando su estructura y las reglas de sintaxis.
2. **Especificación Gramatical**: **Diseñar una gramática libre de contexto** para el lenguaje del caso de estudio, apta para un análisis descendente (evitando recursividad por la izquierda, por ejemplo).
3. **Implementación del DAS**: Utilizar una herramienta como **ANTLR4** para ingresar la gramática y generar el analizador sintáctico.
4. **Pruebas Sintácticas**: Validar el analizador con código fuente de prueba, incluyendo **casos válidos e inválidos** (errores de sintaxis).
5. **Reporte de Análisis**: Elaborar un reporte que detalle la **especificación gramatical**, los **objetivos funcionales** y el uso del **diseño descendente**.
6. **Entrega**: Subir el archivo de la gramática (ej. .g4) y el reporte del análisis a la plataforma.

RESULTADOS ESPERADOS

Un reporte de análisis de caso y una gramática funcional en la herramienta seleccionada, que demuestre la implementación exitosa de un Analizador Sintáctico Descendente (DAS).

ANÁLISIS DE RESULTADOS

Analizar cómo la ambigüedad en la gramática afecta la construcción del DAS y cómo se resuelven los conflictos. Evaluar la robustez del analizador para detectar errores de sintaxis.

CONCLUSIONES Y REFLEXIONES
Concluir sobre la importancia de una gramática bien definida para la fase sintáctica y cómo el enfoque descendente simplifica la traducción. Mencionar la competencia blanda promovida (innovación y trabajo en equipo).

ACTIVIDADES COMPLEMENTARIAS
Realizar una práctica para aplicar los conceptos del análisis semántico básico y el diseño de una tabla de símbolos.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Análisis de caso y código del DAS. Práctica para aplicar los conceptos del análisis semántico básico y el diseño de una tabla de símbolos.
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de análisis de caso.
Formatos de reporte de prácticas	Reporte de análisis de caso y código del DAS.

NOMBRE DE LA PRÁCTICA	Práctica No. 6: Desarrollo de Analizador Semántico con Tabla de Símbolos
COMPETENCIA DE LA PRÁCTICA	Construir y utilizar una tabla de símbolos para almacenar información semántica relevante y aplicar reglas semánticas básicas durante el análisis.

FUNDAMENTO TÉORICO

El análisis semántico verifica el significado y la coherencia del código (ej. tipado, uso de variables). Utiliza una Tabla de Símbolos para almacenar información de los identificadores (nombre, tipo, ámbito) y aplicar Reglas Semánticas para detectar errores (ej. variables no declaradas, incompatibilidad de tipos).

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Computadoras en el aula/laboratorio.
Herramientas de construcción: CUP, ANTLR, JavaCC o Bison/Yacc.
Base de Analizador Sintáctico previamente diseñado.ecursos sobre Análisis Semántico y Tablas de Símbolos

PROCEDIMIENTO O METODOLOGÍA

1. **Diseño de la Tabla de Símbolos**: **En equipos**, definir la estructura de datos para la tabla de símbolos (ej. Hash Map o clases) que almacene el nombre, tipo y ámbito de cada identificador.
2. **Implementación de Reglas de Inserción**: Modificar el analizador sintáctico (o sus acciones) para insertar automáticamente los identificadores en la tabla al momento de su declaración.
3. **Implementación de Reglas Semánticas**: Implementar lógica para aplicar reglas básicas, como: **Verificación de Tipos** en asignaciones o expresiones y **Detección de Uso** de variables no declaradas o duplicidad de declaraciones.
4. **Pruebas Semánticas**: Ejecutar el analizador con casos de prueba que contengan **errores semánticos intencionales** (ej. sumar un entero con una cadena, usar una variable sin declarar) y verificar que se detecten los errores.
5. **Reporte de Práctica**: Documentar el diseño de la tabla, las reglas semánticas aplicadas y adjuntar las **capturas de pantalla o video** de las pruebas.
6. **Entrega**: Subir el código fuente modificado y el reporte de la práctica a la plataforma.

RESULTADOS ESPERADOS

Un Analizador Semántico funcional que utilice una Tabla de Símbolos para verificar la coherencia y el significado del código fuente de prueba.

ANÁLISIS DE RESULTADOS

Analizar la efectividad de la tabla de símbolos para gestionar el ámbito de las variables. Discutir la importancia de la fase semántica para asegurar la ejecución correcta del programa.

CONCLUSIONES Y REFLEXIONES
Concluir sobre la necesidad de la fase semántica para construir compiladores robustos y que generen código intermedio confiable. Mencionar la competencia blanda promovida (innovación y trabajo en equipo).

ACTIVIDADES COMPLEMENTARIAS
Realizar una práctica enfocada en la implementación de reglas semánticas específicas (ej. validación de parámetros de funciones).

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Reporte de práctica Práctica enfocada en la implementación de reglas semánticas
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de reporte de prácticas.
Formatos de reporte de prácticas	Reporte de práctica, código fuente y capturas de pruebas.

NOMBRE DE LA PRÁCTICA	Práctica No. 7: Diseño de Gramática con ANTLR4 para Generación de Código Intermedio
COMPETENCIA DE LA PRÁCTICA	Diseñar una gramática e incluir acciones semánticas en ANTLR4 para la construcción de un Árbol de Sintaxis Abstracta (AST) y la generación de código de tres direcciones.

FUNDAMENTO TÉORICO

La generación de código intermedio (fase de síntesis) utiliza la información del análisis semántico para producir una representación independiente de la máquina (código de tres direcciones, por ejemplo). Esto se logra mediante la traducción dirigida por sintaxis (acciones semánticas embebidas en la gramática).

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Computadoras en el laboratorio.
Herramienta: ANTLR4.
Recursos sobre Traducción Dirigida por Sintaxis y AST

PROCEDIMIENTO O METODOLOGÍA

1. **Diseño de Gramática**: **En equipos**, diseñar una gramática para un subconjunto de lenguaje (ej. expresiones aritméticas y condicionales simples) utilizando el formato **ANTLR4** (.g4).
2. **Construcción del AST**: Configurar la gramática para que ANTLR4 construya automáticamente un **AST (Árbol de Sintaxis Abstracta)** durante el *parsing*.
3. **Acciones Semánticas para Código Intermedio**: Incluir **acciones semánticas** en la gramática para generar **código de tres direcciones** (ej. t1 = a + b) para las operaciones aritméticas al recorrer el AST.
4. **Validación**: Diseñar **casos de prueba** que incluyan expresiones anidadas y estructuras condicionales (si aplica).
5. **Pruebas y Verificación**: Ejecutar ANTLR4 con los casos de prueba y verificar que el código intermedio generado sea el esperado. 6. **Entrega**: Entregar el **archivo .g4** de ANTLR4 y el reporte con los casos de prueba que demuestren la generación de código intermedio.

RESULTADOS ESPERADOS

Un archivo de gramática ANTLR4 con acciones semánticas y un reporte de práctica que muestre la generación exitosa de código de tres direcciones a partir de expresiones de entrada.

ANÁLISIS DE RESULTADOS

Analizar la estructura del AST generado y cómo este facilita el recorrido para la traducción. Evaluar la correcta generación del código de tres direcciones para la aritmética.

CONCLUSIONES Y REFLEXIONES
Concluir sobre la importancia del código intermedio para la optimización y portabilidad del compilador. Mencionar la competencia blanda promovida (análisis y resolución de problemas).

ACTIVIDADES COMPLEMENTARIAS
Realizar una práctica para implementar una tabla de símbolos jerárquica con Bison/yacc para el manejo de ámbitos durante la generación de código.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Reporte de prácticas
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de reporte de prácticas.
Formatos de reporte de prácticas	Archivo .g4 y reporte con capturas de pantalla/video de la ejecución

NOMBRE DE LA PRÁCTICA	Práctica No. 8: Integración de Herramientas (jflex + CUP/Bison) para un Compilador Completo
COMPETENCIA DE LA PRÁCTICA	Construir un flujo completo de análisis léxico-sintáctico-semántico con generación de código intermedio integrando diferentes herramientas (jflex + CUP/Bison).

FUNDAMENTO TÉORICO

Esta práctica integra las fases clave de un compilador: Análisis Léxico (jflex), Análisis Sintáctico (CUP/Bison), Análisis Semántico (Tabla de Símbolos) y Generación de Código Intermedio (Traducción Dirigida por Sintaxis). Representa la culminación de los conocimientos del curso.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Computadoras en el aula/laboratorio.
Herramientas de construcción: jflex (lexer) + CUP/Bison (parser).
Recursos sobre compiladores y el uso de las herramientas integradas.

PROCEDIMIENTO O METODOLOGÍA

1. **Definición del Lenguaje**: **En equipos**, definir un lenguaje que incluya declaraciones de variables, funciones, estructuras de control (if/while) y expresiones complejas.
2. **Fase Léxica (jflex)**: Implementar el analizador léxico que genere los tokens correspondientes al lenguaje definido.
3. **Fase Sintáctica/Semántica (CUP/Bison)**: Implementar el analizador sintáctico. Integrar el análisis semántico con la **tabla de símbolos** jerárquica para verificar tipos y ámbitos.
4. **Generación de Código Intermedio**: Modificar el parser (CUP/Bison) para que, durante el análisis, genere **código intermedio** (ej. código de tres direcciones) para las expresiones y estructuras de control.
5. **Pruebas y Validación**: Evaluar el flujo completo con **programas de ejemplo** del lenguaje diseñado. Se debe probar la detección de errores léxicos, sintácticos y semánticos, así como la correcta generación de código intermedio.
6. **Entrega**: Entregar el **compilador funcional** (código fuente) junto con un **informe de diseño** y pruebas detalladas.

RESULTADOS ESPERADOS

Un compilador funcional que demuestre la integración de todas las fases del proceso de compilación y que genere código intermedio para programas de entrada.

ANÁLISIS DE RESULTADOS

Analizar la eficiencia del flujo de trabajo integrado entre jflex y CUP/Bison. Evaluar cómo la implementación de la tabla de símbolos y la generación de código intermedio contribuyen a la solución de la problemática definida.

CONCLUSIONES Y REFLEXIONES
Concluir sobre la complejidad de construir un compilador desde cero y cómo las herramientas automatizadas facilitan el proceso. Mencionar la aplicación de buenas prácticas de desarrollo de software y la competencia blanda promovida (análisis y resolución de problemas).

ACTIVIDADES COMPLEMENTARIAS
Realizar una práctica de optimización de código intermedio (ej. constant folding) a partir de un AST con CUP.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Código fuente del compilador integrado.
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de proyecto integrador.
Formatos de reporte de prácticas	Código fuente del compilador integrado, informe de diseño y pruebas.

FUENTES DE INFORMACIÓN

Actas de JENUI - Edición integral. (s. f.), (2003), Del propósito de la materia de compiladores en la formación del ingeniero informático.

Alenda, A. G. (2002). Diseño de compiladores.

Alfonseca Cubero, E. (2007). Teoría de autómatas y lenguajes formales. McGraw-Hill España.

Carranza Sahagún, D. U. (Coord.). (2024). Compiladores: fases de análisis: (1 ed.). Editorial Transdigital.

Fernando, Irving & Alvarado Asensio, Irving. (2024). LA EVOLUCIÓN DE LOS COMPILADORES Y SU IMPACTO EN LA PROGRAMACIÓN MODERNA.

GeeksforGeeks. (s. f.). Regular Expressions, Regular Grammar and Regular Languages.

Gomez, Cristian. (2023). La evolución de los compiladores.

Herrera Hernández, E. (2006). Compilación II. Editorial Félix Varela.

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2008). Teoría de autómatas, lenguajes y computación 3/E. Pearson Educación.

Java a Tope: Traductores Y Compiladores Con Lex/yacc, Jflex/cup Y Javacc. (s/f). Sergio Gálvez Rojas.

Jiménez Millán, J. A. (2014). Compiladores y procesadores de lenguajes. Servicio de Publicaciones de la Universidad de Cádiz.

Libretexts. (2021, 1 agosto). 3: Regular Expressions and FSA's. Engineering Libre Texts.

Lovelle, J. M. C. (2003). Lenguajes, gramáticas y autómatas en procesadores de lenguaje.

Málaga, E. J. (2008a). Teorías de Autómatas y Lenguajes Formales.

Martínez López, F. (2015). Teoría, diseño e implementación de compiladores de Lenguajes. RA-MA Editorial.

Mayol i Badía, A. (2009). Autómatas programables: (ed.). Marcombo.

Mendoza García, M. G. y Mendoza García, M. G. (2015). Teoría de autómatas: un enfoque práctico. Pearson Educación.

Muralles, Damaris. (2023). LA PROGRAMACIÓN SIN LOS COMPILADORES.

Pahade, P. (2019). INTRODUCTION TO COMPILER AND ITS PHASES.

Rafael, R. A. R., & Recto, K. H. A. (s. f.). Influence of Language Evolution and Compiler Advances on Program Creation: Implications to Electronics Engineering Education. Archium Ateneo.

Sánchez Dueñas, G. & Valverde Andreu, J. A. (2022). Compiladores e intérpretes: un enfoque pragmático: (1 ed.). Ediciones Díaz de Santos.

Sassi, R. B. (2024, 10 septiembre). Compiler vs. Interpreter in Programming. Built In.

Vega Castro, R. A. (2008). Compilador de pseudocódigo como herramienta para el aprendizaje en la construcción de algoritmos: (1 ed.). Bubok Publishing S.L.


UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu