Nombre de la Asignatura: **PROCESADORES DE LENGUAJES II** Código: **505**

Créditos Teóricos: **4** Créditos Prácticos: **1.5** Créditos Totales:**5.5**

Nombre del Departamento: **Estadística, Investigación Operativa y Computación.**

Profesor que imparte la asignatura: **Coromoto León Hernández** Teléfono: **922 31 81 80**

Horario de Tutorías: **Lunes, Miércoles: 10:30-13:30** e-mail: **coro@csi.ull.es**

Horario de Clases: **Lunes 15:00-15:50; Martes 15:00-15:50, 16:00 - 16:50**

Número de Horas Prácticas: **1 (Sala Unix)** Número de Grupos: **3**

Horario de Prácticas: **Jueves: 14:00-15:00; 15:00-16:00; 16:00-17:00**

Método de Evaluación:

La nota de la asignatura está repartida de la siguiente manera:

80% Teoría + 20% Prácticas

La calificación obtenida en el examen teórico constituye el 80% de la nota final de la asignatura. Semanalmente se evalúan los ejercicios prácticos en el laboratorio, constituyendo la totalidad de los mismos el 20% de la nota final de la asignatura. Si no se supera la realización del **75%** de los Ejercicios Prácticos, se cuenta con un examen práctico final. Cada ejercicio práctico tendrá asociado un factor de ponderación que se publicará junto con la fecha de cada uno.

Programa de Teoría:

1.- **Introducción.**
Generalidades. Conceptos de Lenguajes, sintaxis, semántica. Esquema de un compilador. Definiciones dirigidas por la sintaxis. Evaluación ascendente de definiciones con atributos sintetizados. Definiciones con atributos por la izquierda. Traducción descendente. Evaluación ascendente de atributos heredados.

2.- **Generación de Código Intermedio**.
Lenguajes intermedios. Declaraciones. Proposiciones de asignación. Expresiones booleanas. Proposiciones case. Relleno de retroceso. Llamadas a procedimientos.

3.- **Generación de Código.**
Bloques básicos y grafos de flujo. Distribución y asignación de registros. Optimización mediante mirilla. Generadores de generadores de código.

4.- **Optimización de Código.**
Optimización de bloques básicos. Bucles en los grafos de flujo. Introducción al análisis global del flujo de datos. Solución iterativa de las ecuaciones de flujo de control. Transformaciones para mejorar el código. Tratamiento con alias. Análisis de flujo de datos de grafos de flujo estructurados. Algoritmos eficientes para el flujo de datos.

Programa de Prácticas:

1.- **Título :** ***Asignación de Memoria en Sample C.***
Objetivo : Implementar la asignación de memoria en el compilador de *Sample C*.

2.- **Título :** ***Lenguajes Intermedios: P-código para C.***
Objetivo : Implementación de un intérprete de código intermedio.

3.- **Título :** ***Lenguajes Intermedios: Árboles.***
Objetivo : Implementación de un intérprete de código intermedio para representaciones gráficas.

4.- **Título :** ***Lenguajes Intermedios: Grafos dirigidos acíclicos.***
Objetivo : Optimización del intérprete de código intermedio implementado en la práctica anterior.

5.- **Título :** ***Generación de código intermedio (Expresiones).***
Objetivo : Partiendo del front-end de un compilador para un lenguaje sencillo, ampliarlo con la generación de un código intermedio para el tratamiento de expresiones.

6.- **Título :** ***Generación de código intermedio (Tipos estructurados).***
Objetivo : Partiendo del front-end de un compilador para un lenguaje sencillo, ampliarlo con la generación de un código intermedio para el tratamiento de arrays y registros.

7.- **Título :** ***Generación de código intermedio (Flujo de Control).***
Objetivo : Resolver el problemas de las referencias hacia delante usando una tabla o bien la técnica del back-patching.

8.- **Título :** ***Generación de código intermedio (Subprogramas).***
Objetivo : Admisión de la declaración en invocación de funciones, generando la representación intermedia en cada caso.

9.- **Título :** ***Optimización de código (Bloques Básicos).***
Objetivo : Implementar técnicas de optimización en el código para una máquina hipotética.

10.- **Título :** ***Optimización de código (Optimizaciones Locales y de bucles).***
Objetivo : Implementar técnicas de optimización en el código para una máquina hipotética.

11.- **Título :** ***Generación de código (Expresiones, Sentencias y Subprogramas).***
Objetivo : Partiendo de un código intermedio implementar la generación de código para expresiones, sentencias y subprogramas en un TRANSPUTER.

Bibliografía:

[1] Aho, A.; Sethi, R.; Ullman, J., "Compilers.Principles, Techniques and Tools", Addison-Wesley Publishing company, 1986.

[2] Bennet, J. P., "Introduction to compiling techniques. A First Course using ANSI C. LEX and YACC". McGraw-Hill, 1990.

[3] Fischer, C.N., LeBlanc, R.; "Crafting a compiler with C". The Benjamin/Cummings Publising Company, Inc. 1991.

[4] INMOS Limited; "Transputer Instruction Set. A compiler writer's guide", Prentice Hall, 1988.

[5] Levine, J.; Mason, T.; Brown, D., "LEX & YACC", O'Reilly & Associates, Inc. 1990.

[6] León, C., "Prácticas de Compiladores en C y Pascal", Colección Textos Universitarios. Gobierno de Canarias. 1997.

[7] Schreiner, A.; Friedman, Jr. H, "Introduction to compiler construction wiht Unix". Prentice Hall, 1985.