

INDICE

Prefacio	XV
1 Introducción	1
1.1. ¿Por qué estudiar diseño de compiladores?	
1.1.1. El diseño de compiladores, campo de éxito	5
1.1.2. Aplicación de las técnicas de diseño de compiladores	7
1.1.3. Los compiladores contienen algoritmos de uso general	
1.2. Un compilador/interprete modular tradicional sencillo	8
1.2.1. El árbol sintáctico abstracto	
1.2.2. Estructura del compilador de demostración	
1.2.3. El lenguaje del compilador de demostración	10
1.2.4. Análisis léxico del comprador de demostración	12
1.2.5. Análisis sintácticos par el compilador de demostración	14
1.2.6. Gestión del contexto en el compilador de demostración	17
1.2.7. Generación de código en el compilador de demostración	18
1.2.8. Back-end de interpretación para el compilador de demostración	19
1.3. Estructura de un compilador mas realista	
1.3.1. La estructura	20
1.3.2. Sistema en tempo de ejecución	22
1.3.3. Algunos atajos	
1.4. Arquitectura de compiladores	23
1.4.1. Anchura del compilador	
1.4.2. ¿Quién es el jefe?	25
1.5. Propiedades de un buen compilador	26
1.6. Portabilidad y cambio de destino	28
1.7. Lugar y utilidad de las optimizaciones	
1.8. Breve historia del diseño de compiladores	29
18.1. 1945-1960: generación de código	
1.8.2. 1960-1975: análisis	
1.8.3. 1975-actualidad: generación de código y optimización de código	
paradigmas	30
1.9. Gramáticas	
1.9.1. Forma de una gramática	
1.9.2. El proceso de producción	31
1.9.3. Formas extendidas de una gramática	32
1.9.4. Propiedades de las gramáticas	
1.9.5. Formalismo de las gramáticas	34
1.10. Algoritmos de clausura	36
1.10.1. Una implementación iterativa del algoritmo de clausura	40
1.11. Pseudocódigo	41
1.12. Conclusión	
Resumen	42
2 Del texto de programa al árbol sintáctico abstracto	46
2.1. Del texto de programa al los tokens- la estructura léxica	
2.1.1. Lectura del texto del programa	50
2.1.2. Análisis léxico frente análisis sintáctico	51
2.1.3. Expresiones regulares y descripciones regulares	52
2.1.4. Análisis léxico	54

2.1.5. Implementación manual de un analizador léxico	55
2.1.6. Generación automática de un analizador léxico	62
2.1.7. Comprensión de la tabla de transición	79
2.1.8. La gestión de errores en los analizadores léxicos	85
2.1.9. Lex-un analizador léxico clásico	86
2.1.10. Identificación léxica de tokens	88
2.1.11. Tablas de símbolos	90
2.1.12. Procesamiento de macros e inclusión de archivos	94
2.1.13. Conclusión	100
2.2. De los tokens al árbol sintáctico- la sintaxis	101
2.2.1. Dos clases de metodologías de análisis sintáctico	102
2.2.2. Detección de errores y recuperación de errores	105
2.2.3. Implementación manual de un analizador descendente	107
2.2.4. Generación automática de un analizador descendente	109
2.2.5. Generación automática de un analizador ascendente	138
2.3. Conclusión	167
Resumen	168
Lecturas complementarias	
Ejercicios	171
3 Anotación del AST el contexto	180
3.1. Gramáticas de atributos	181
3.1.1. Grafos de dependencias	185
3.1.2. Evaluación de atributos	186
3.1.3. Tratamiento de ciclos	194
3.1.4. Asignación de atributos	201
3.1.5. Gramáticas de atributos multivisita	202
3.1.6. Resumen de los tipos de gramáticas de atributos	
3.1.7. Gramáticas L-atribuidas	213
3.1.8. Gramáticas S-atribuidas	
3.1.9. Equivalencia de gramáticas L-atribuidas	218
3.1.10. Notaciones de gramáticas y gramáticas de atributos extendidos	219
3.1.11. Conclusión	220
3.2. Métodos manuales	
3.2.1. Enhebrado del AST	221
3.2.2. Interpretación simbólica	226
3.2.3. Ecuaciones de flujo de datos	234
3.2.4. Análisis de flujo de datos	241
3.2.5. Programación de la información aguas arriba-análisis en vivo	242
3.2.6. Interpretación simbólica y ecuaciones	
3.3. Conclusión	247
Resumen	249
Lecturas complementarias	252
Ejercicios	253
4 Procesado del código intermedio	258
4.1. Interpretación	
4.1.1. Interpretación recursiva	260
4.1.2. Interpretación iterativa	264
4.2. Generación de código	267
4.2.1. Evitar la generación de código por completo	272

4.2.2. El punto de partida	273
4.2.3. Generación de código trivial	274
4.2.4. Generación de código sencilla	279
4.2.5. Generación de código para bloques básicos	295
4.2.6. Generación de código BURS y programación dinámica	311
4.2.7. Asignación de registros por coloreado de grafos	330
4.2.8. Supercompilación	335
4.2.9. Evaluaron de técnicas de generación de código	336
4.2.10. Depuración de optimizadotes de código	337
4.2.11. Preprocesado del código intermedio	338
4.2.12. Postprocesado del código destino	342
4.2.13. Generación de código maquina	345
4.3. Ensambladores, enlazadores y cargadores	346
4.3.1. Notas sobre el diseño de ensambladores	349
4.3.2. Notas sobre el diseño de enlazadores	351
4.4. Conclusión	352
Resumen	353
Lecturas complementarias	
Ejercicios	358
5 Gestión de memoria	365
5.1. Reserva de datos con desalojo explicito	367
5.1.1. Reserva de memoria básica	368
5.1.2. Listas enlazadas	372
5.1.3. Arrays extensibles	373
5.2. Reserva de datos con desalojo implícito	375
5.2.1. Algoritmos básicos de recolección de basura	376
5.2.2. Preparación del terreno	377
5.2.3. Cuenta de referencias	383
5.2.4. Marcar y escanear	386
5.2.5. Copia de dos espacios	392
5.2.6. Compactación	394
5.2.7. Recolección de basura generacional	396
5.3. Conclusión	
Resumen	397
Lecturas complementarias	400
Ejercicios	401
Programas imperativos y programas orientados a objetos	403
6.1. Gestión del contexto	405
6.1.1. Identificación	406
6.1.2. Comprobación de tipos	413
6.1.3. Conclusión	422
6.2. Gestión y representación de los datos del lenguaje fuente	
6.2.1. Tipos básicos	
6.2.2. Tipos de enumeración	423
6.2.3. Tipos puntero	
6.2.4. Tipos registro	427
6.2.5. Tipos unión	428
6.2.6. Tipos array	429
6.2.7. Tipos conjunto	432

6.2.8. Tipos rutina	
6.2.9. Tipos objeto	
6.2.10. Tipos interfaz	441
6.3. Rutinas y su activación	
6.3.1. Registros de activación	442
6.3.2. Rutinas	445
6.3.3 Operaciones sobre rutinas	446
6.3.4. Rutinas no anidadas	449
6.3.5. Rutinas anidadas	450
6.3.6. Lambda lifting	458
6.3.7. Iteradores y co-rutinas	
6.4. Generación de código para las sentencias de flujo de control	460
6.4.1. Flujo de control local	461
6.4.2. Invocación de rutinas	470
6.4.3. Gestión de errores en tiempo de ejecución	477
6.5. Generación de código para módulos	480
6.5.1. Generación de nombres	
6.5.2. Inicialización de módulos	481
6.5.3. Generación de código para unidades genéricas	482
6.6. Conclusión	483
Resumen	484
Lecturas complementarias	
Ejercicios	487
7 Programas funcionales	494
7.1. Una pequeña visita por Haskell	495
7.1.1. Regla de fuera de juego	496
7.1.2. Listas	497
7.1.3. Listas por comprensión	
7.1.4. Búsqueda de patrones	498
7.1.5. Tipado polimórfico	499
7.1.6. Transparencia referencial	500
7.1.7. Funciones de orden superior	501
7.1.8. Evaluación perezosa	502
7.2. Compilando lenguajes funcionales	503
7.2.1. El núcleo funcional	505
7.3. Comprobación de tipos polimórficos	506
7.3.1. Aplicación de funciones polimórficas	507
7.4. Eliminación del azúcar sintáctico	
7.4.1. Traducción de listas	508
7.4.2. Traducción de búsqueda	
7.4.3. Traducción de listas por compresión	511
7.4.4. Traducción de listas anidadas	513
7.5. Reducción de grafos	514
7.5.1. Orden de reducción	519
7.5.2. El motor de reducción	521
7.6. Generación de código para núcleos	524
7.6.1. Evitar la construcción de algunas espinas de aplicaciones	526
7.7. Optimización del núcleo funcional	528
7.7.1. Análisis de argumentos estrictos	529

7.7.2. Análisis de encaje	534
7.7.3. Llamadas desde la cola	535
7.7.4. La transformación del acumulador	537
7.7.5. Limitaciones	
7.8. Manipulación avanzada de grafos	539
7.8.1. Nodos de longitud variable	
7.8.2. Etiquetado de punteros	
7.8.3. Asignación agregada de nodos	540
7.8.4. Nodos vector apply	
7.9. Conclusión	
Resumen	541
Lecturas complementarias	
Ejercicios	544
8 Programas lógicos	548
8.1.1 El modelo de programación lógica	
8.1.1. Los bloques de constitutivos	550
8.1.2. El mecanismo de inferencia	551
8.2. El modelo general de implementación, interpretado	552
8.2.1. Instrucciones del interprete	554
8.2.2 Evitar listas de metas redundantes	
8.2.3. Evitar copiar las colas de las listas de metas	557
8.3. La unificación	
8.3.1. Unificación de estructuras, listas y conjuntos	558
8.3.2. La implementación de la unificación	560
8.3.3. Unificación de dos variables no ligadas	562
8.3.4. Conclusión	564
8.4. El modelo de implementación general, compilado	565
8.4.1. Procedimientos de lista	566
8.4.2. Búsqueda y unificación de cláusulas compilada	568
8.4.3. Selección de cláusulas optimizada en WAN	572
8.4.4. Implementación el mecanismo corte	576
8.4.5. Implementación de los predicados assert y retract	578
8.5. Código compilado para la unificación	582
8.5.1. Instrucciones de unificación en WAN	583
8.5.2. Derivación de una instrucción de unificación mediante evaluación parcial manual	585
8.5.3. Unificación de estructuras en WAN	588
8.5.4. Una optimización: Modo lectura/escritura	591
8.5.5. Mas optimizaciones de unificación en WAN	593
8.5.6. Conclusión	
Resumen	596
Lecturas complementarias	
Ejercicios	598
9 Programas paralelos y distribuidos	601
9.1. Modelos de programación paralela	
9.1.1. Variables compartidas y monitores	604
9.1.2. Modelos de paso de mensajes	605
9.1.3. Lenguajes orientados a objetos	607
9.1.4. El espacio de tuplas Linda	608

9.1.5. Lenguajes de datos paralelos	609
9.2. Procesos e hilos	611
9.3. Variables compartidas	
9.3.1. Cerrojos	612
9.3.2. Monitores	
9.4. Paso de mensajes	613
9.4.1. Localizar el receptor	
9.4.2. Ordenación	615
9.4.3. Comprobación de tipos de mensajes	
9.4.4. Selección de mensajes	617
9.5. lenguajes paralelos orientados a objetos	
9.5.1. Localización de objetos	618
9.5.2. Migración de objetos	
9.5.3. Replicación de objetos	
9.6. El espacio de tuplas	
9.6.1. Evitando la sobrecarga del direccionamiento asociativo	621
9.6.2. Implementaciones distribuidas del espacio de tuplas	624
9.7. Paralelización automática	626
9.7.1. Explotando el paralelismo automáticamente	627
9.7.2. Dependencias de datos	628
9.7.3. Transformaciones de bucles	630
9.7.4. Paralelización automática par maquinas de memoria distribuida	631
9.8 Conclusión	
Resumen	634
Lecturas complementarias	
Ejercicios	636
Apéndice A-un compilador/interprete orientado a objetos	
A.1 Clases y métodos	639
A.2 Un sencillo compilador orientado a objetos	
A.3 Análisis sintáctico a objetos	641
Evaluación	
Ejercicios	649
Respuestas a los ejercicios	650
Referencias	663
Índice	671