مرور مطالب درس طراحی کامپایلر

در ارزشیابی نهایی درس طراحی کامپایلر، برخی از سؤالات به مطالب ارائه شده در نیمه‌ی اول (به خصوص روش‌های تجزیه) و برخی بـه مطالب ارائه شده در نیمه‌ی دوم اختـصاص داده خواهنـد شد. سرفـصل مطالب گذرانـده شـده در نیمه‌ی دوم درس در ادامه‌ی این مستند فهرست می‌شوند.

عمل‌ها (کنش‌ها) و مقادیر مفهومی Semantic Actions (Rules) and Values 1:5.0 1:5.1 1:6.4 1:6.3 2:4.1

استفاده برای ارزیابی نتیجه‌ی عبارت‌ها

درخت مجرد (Abstract Syntax Tree)

استفاده برای بررسی نوع داده (Type-checking)

انواع داده و تبدیل‌ها (Type conversions)

کد میانی Intermediate Code 1:6.2 1:6.4

کد سه-آدرسه (Three-address codes) و انواع آدرس‌های آن

اصول استفاده از کنش‌های مفهومی برای تولید کد میانی

محیط زمان اجرا و مدیریت حافظه Runtime environments 1:7.1 1:7.2 2:6

مدیریت ایستا و پویای حافظه

مفهوم فعال‌سازی رویه‌ها (Procedure activation) و درخت‌های فعال‌سازی

سازماندهی پشته و «Activation records»

جمع‌آوری زباله Garbage Collection 1:7.5 2:13

مفهوم و اهداف

اصول روش «Reference counting» و مشکل دورها

اصول روش‌های «Trace-based» و مفهوم مجموعه‌ی ریشه (Root set)

بلوک‌های پایه Basic Blocks 1:8.4 2:8

تشخیص بلوک‌های پایه

زنده‌بودن متغیرها و استفاده‌ی بعدی آنها (Liveness and next-use)

گراف‌های جریان Flow graphs 1:8.4 2:8

ساخت گراف جریان از روی بلوک‌های پایه

تشخیص حلقه‌ها

تحلیل زنده‌بودن سراسری Global Liveness Analysis 1:9.2.5 2:10

تخصیص رجیسترها و تولید کد نهایی Register Allocation 1:8.8 2:11 2:9.2

تفاوت‌های معماری‌های RISC و CISC

مسئله‌ی انتخاب دستور (Instruction selection)

تخصیص رجیستر در هنگام تولید کد نهایی یک بلوک پایه

تخصیص رجیستر سراسری با رنگ‌آمیزی گراف

بهینه‌سازی‌های محلی و سراسری Local and Global Optimizations 1:8.5 1:9.1

آشنایی با حذف کد مرده، انتشار ثوابت، تشخیص عبارت‌های مشابه

چند سؤال نمونه از الگوریتم‌های نیمه‌ی دوم

۱) بلوک‌های پایه‌ی (Basic Blocks) کد میانی زیر را بدست آورید و از روی آن گراف جریان (Flow graph) آن را بکشید و حلقه‌های آن را تشخیص دهید. در هر بلوک زنده‌بودن (Liveness) و استفاده‌ی بعدی (Next-use) متغیرها را بدست آورید. با استفاده از تحلیل زنده بودن (Liveness Analysis) سراسری، گراف تداخل رجیستر (Register Interference Graph) را بکشید و با استفاده از رنگ‌آمیزی گراف، سه رجیستر را به متغیرها اختصاص دهید.

۲) درخت فعال‌سازی (Activation Tree) را برای شبه‌کد زیر بکشید.

۳) در شکل زیر اشیاء (Objects) موجود در حافظه با یک مستطیل نمایش داده شده‌اند و یک یال جهت‌دار شی A را به شی B وصل می‌کند اگر A به B اشاره کند (یا یک Reference از آن را نگه دارد). اشیائی که با استفاده از یک نام در برنامه قابل دسترسی هستند (مجموعه‌ی ریشه یا «Root set») در یک دایره قرار گرفته‌اند. تعداد Reference-ها را برای هر یک از اشیاء محاسبه کنید و با استفاده از الگوریتم Reference-counting، زباله‌ها را تشخیص دهید. با استفاده از الگوریتم Mark-and-Sweep (پیمایش اشاره‌گرها از اشیاء مجموعه‌ی پایه)، زباله‌ها را تشخیص دهید.

یادآوری: اشاره از مجموعه‌ی پایه نیز یک Reference شمرده می‌شود. در روش Reference-counting اشیائی که تعداد اشاره‌ها به آنها صفر است، آزاد می‌شوند. پس از آزاد کردن یک شی، تعداد اشاره‌ها به اشیائی که به آنها اشاره می‌کند به روز می‌شود و ممکن است زباله‌های جدیدی شناسایی شوند.