Kapitola 18

Programovací jazyky a překladače

18.1 Otázky

Následující seznam otázek shnuje mou představu o tom, co je z okruhu Programovací jazyky a překladače ke státnicím potřeba umět. Tato představa nebyla konzultována s vyučujícím(i).

Struktura kompilátoru a navazujících nástrojů (linkery, loadery, debuggery, knihovny, preprocesory)

- schéma překladače
- popis linkeru
- popis loaderu (Unix, Windows)
- statické knihovny, dynamické knihovny, implmentace
- popis C preprocesoru

Konečné automaty a lexikální analýza

- úkol lexikální analýzy
- výstup lexikální analýzy, varianty, problémy
- typické způsoby implementace
- lex, flex

Syntaktická analýza — LL, LR techniky

- úkol syntaktické analýzy
- výstup syntaktické analýzy (vč. možností při analýze shora dolů a zdola nahoru)
- pojmy: gramatika, bezkontextová gramatika, derivace, levá/pravá derivace
- definice: FIRST a FOLLOW
- analýza shora dolů, vztah s LL, definice a popis LL, pojmy rozepsání a krácení
- rozdíly mezi slabou a silnou LL
- implementace analýzy shora dolů rekurzivním sestupem
- analýza zdola nahoru, vztah s LR, definice a popis LR, pojmy shift a reduction
- rozdíly mezi LR, SLR a LALR
- konstrukce LR, SLR a LALR automatu
- yacc, bison
- gramatiky s regulární pravou stranou

Syntaxí řízený překlad a atributové gramatiky

- derivační strom vs. AST
- definice: AG, Attr, Syn, Inh, normální forma, atributový výskyt, In, Out
- definice: acyklické AG, k-průchodové AG, jednoduše k-průchodové AG, zleva-doprava k-průchodové AG, jednoduše zleva-doprava k-průchodové AG
- algoritmus konstrukce průchodů jednoduše zleva-doprava k-průchodovou AG

Reprezentace programu mezikódem

- důvody reprezentace mezikódem
- pojem základní blok
- formáty mezikódu: sekvenční (2-adresový, 3-adresový (trojice, čtveřice), prefixová/postfixová notace), graf základních bloků, strom, DAG
- operandy mezikódu
- SSA
- funkce u trojic a SSA
- generování mezikódu (použití AG, zisk z proházení operací, problém LHS a RHS u přiřazení a jeho řešení, synchronizace u DAGů)

Překlad výrazů a programových struktur

- preklad programove struktur — control-flow, tedy if/when/for/goto pripadne try/catch
- vyrazy — zakladni princip vyhodnocovani, CSE, eventualne eliminace duplicitnich read/write operaci.

Rozsahy platnosti proměnných, aktivační záznamy, implementace vnořených procedur, volací konvence

- obsah aktivačního záznamu
- způsoby implementace vnořených procedur (odkaz na všechny záznamy, odkaz na předchozí záznam, displej) a jejich vlastnosti (režie volání, přístupu k proměnným a prostorová náročnost)
- implementace procedurálních parametrů v Pascalu
- volací konvence: co vše zahrnují (registry vs. zásobník, zachovávání registrů, pořadí předávání parametrů, návratová hodnota, úklid zásobníku)
- popis cdecl, stdcall, fastcall

Vliv architektury počítače na generování kódu a optimalizaci

Zdroje

- Wikipedia: Microarchitecture

Obsah otázky

- registry (počet, typy, způsob přístupu, (ne)ortogonalita)
- instrukční sada (RISC, CISC, (ne)ortogonalita)
- pipelining
- superskalarita
- out-of-order execution
- spekulativní vykonávání instrukcí
- SIMD
- práce s pamětí (heap, stack), zarovnávání

Metody generování kódu, přidělování registrů, optimalizace

Zdroje

- Wikipedia: Category: Compiler optimizations

Obsah otázky

- klasifikace základních bloků podle rozsahu platnosti, výpočet liveBegin a liveEnd, určení kolizí
- výběr a uspořádání instrukcí (1:1, 1:n, m:1, m:n), pojem generator-generator, řazení instrukcí
- objekty umisťované do registrů
- algoritmus obarvování grafu
- optimalizace: důvody, čas vs. prostor, úroveň, požadované vlastnosti
- druhy optimalizací: lokální, globální, v rámci programu
- lokální optimalizace: CSE, copy propagation, dead-code elimination, constant folding, algebraické operace
- globální optimalizace: CSE, copy propagation, dead-code elimination, optimalizace cyklů (přesun invariantního kódu, redukce síly operace, odstranění indukční proměnné)
- paralelizace a vektorizace
- profilem řízená optimalizace

Podpora kompilátorů pro synchronizační primitiva, vlákna

Zdroje

- Using wait(), notify() and notifyAll()

Obsah otázky

- Java: kritické sekce, monitory (synchronized), wait, notify
- Java: vytvoření vlákna, paměťový model, TLS, podpora v knihovnách (thread (un)safe)
- volatile proměnné
- podpora knihovnami při absenci podpory kompilátoru

Objektově orientované jazyky a principy jejich implementace

Zdroje

- Memory Layout for Multiple and Virtual Inheritance
- Compilation of object oriented languages

Obsah otázky

- principy OO: zapouzdření, dědičnost a polymorfismus
- class-based languages vs. prototype-based languages
- OO podle Smalltalk: zasílání zpráv
- implementace dědičnosti: tabulka virtuálních funkcí, volání virtuálních funkcí, princip pozdní vazby, layout objektů při jednonásobné a vícenásobné dědičnosti, diamond problem, virtuální dědičnost, RTTI a jeho implementace (typeid, dynamic_cast)

Překladače vs. interpretry, skriptovací jazyky

- rozdíly mezi překladačem a interpretrem
- výhody/nevýhody překladačů a interpretrů
- typické příklady použití překladačů a interpretrů
- bytecode, virtuální stroj (zásobník vs. registry), JIT, příklady (Java, Lua)
- optimalizační techniky u interpreterů: method caching, direct threading, value tagging
- typické vlastnosti skriptovacích jazyků (dynamické typování, reflexivita, dynamičnost)
- typické použití skriptovacích jazyků (skripty, web,...), důvody

18.2 Materiály

- Web předmětu Principy překladačů
- Web předmětu Konstrukce překladačů