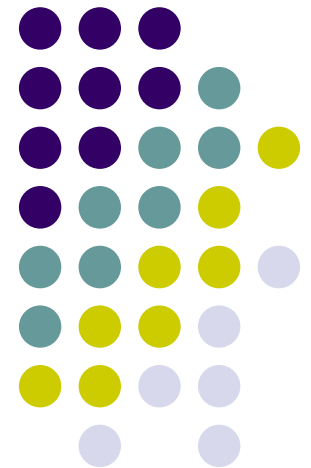
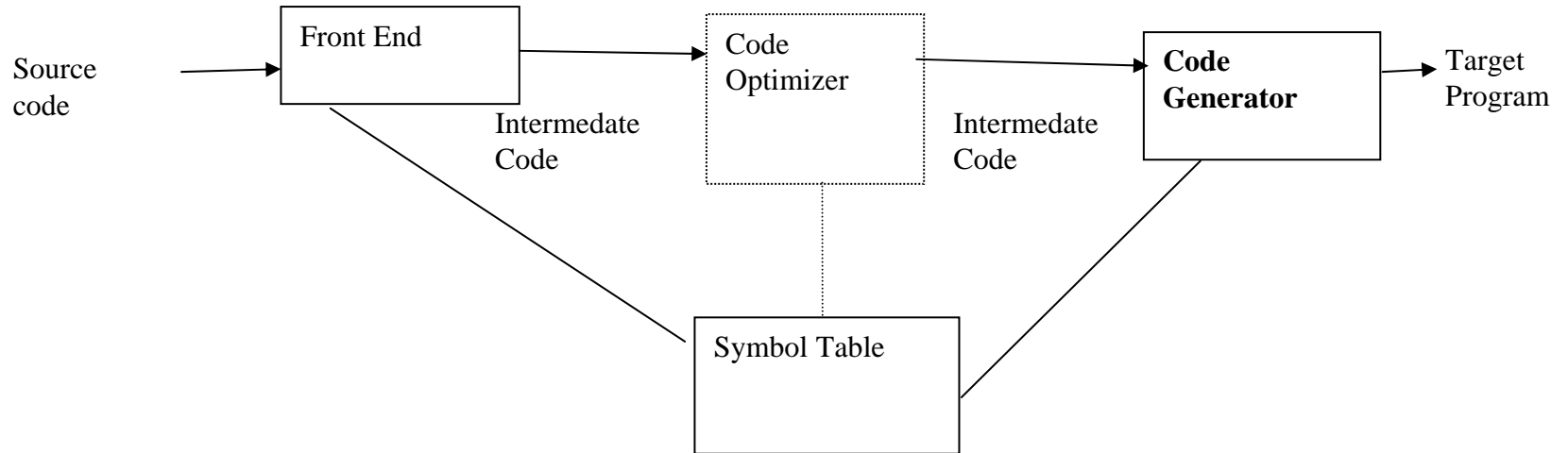


Teknik Kompiler 12

oleh: **antonius rachmat c,**
s.kom



Code Generator

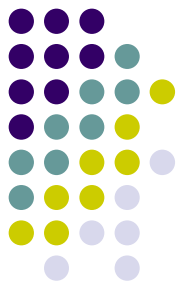


Code Generation



- Persyaratan Code Generation:
 - Output kode harus benar dan berkualitas tinggi, menggunakan resources dan target machine secara efektif dan efisien
 - Prakteknya proses ini tidak dapat diprediksi
 - Bisa menggunakan teknik heuristik

Isu-isu pada Code Generation



- Target Program
 - Output dari code generator adalah salah satu dari:
 - Absolute Machine Language
 - Mempunyai keuntungan dapat ditempatkan pada tempat yang fixed di dalam memori dan langsung dapat dieksekusi, program kecil.
 - Relocatable Machine Language
 - Obyek module dapat dikompilasi secara terpisah, satu set relocatable obyek modul dapat dilink bersama-sama dan dapat diload oleh linking loader untuk dieksekusi.
 - Assembly Language
 - Mempermudah proses code generation

Isu-isu Code Generation



- Apa yang harus diketahui tentang prosesor target?
 - Untuk semua perintah yang mengandung operasi, kita harus tahu mana register yang diakses dari operand-nya
 - Perintah-perintah yang bisa digunakan di mesin tersebut
 - Bagaimana cara membaca input dan output dari mesin? (I/O)
 - Bagaimana pengalokasian dan manajemen resourcenya (resource usage)
 - Seberapa kecepatan input/output dan perintah-perintah yang digunakan latest write, fastest write, latest read, fastest write (latency)
 - Exception handler
 - Interrupt handler

Isu-isu code generation



- Input untuk code generator
 - Input adalah IR bersama dengan simbol-simbol dari tabel simbol yang digunakan untuk menentukan runtime address dari data obyek yang ditunjukkan oleh nama-nama dalam IR
 - Diasumsikan sudah dilakukan type checking, type conversion, dan error handling (akan dibahas)

Isu-isu code generation



- Memory management
 - Memetakan nama di dalam source program ke dalam addressnya.
 - Semua nama-nama dalam IR diasumsikan merefer pada tabel simbol untuk nama tersebut
 - Tipe dalam deklarasi menentukan lebar data (jumlah storage) yang diperlukan untuk nama yang dideklarasikan.



Isu-isu code generation

- Instruction Selection
 - Harus uniform
 - Memperhatikan instruction speed dan machine idiom
- Register Allocation
 - mengalokasikan mana register-register yang akan dipakai (dialokasikan)
 - Untuk nilai-nilai yang sering dipakai, gunakan Global Register Allocation
 - Lebih bagus jika lebih banyak menggunakan register untuk operand daripada menggunakan operand di dalam memory.
 - Selama register allocation dipilih variabel-variabel mana yang akan menetap pada register.
 - Selama register assignment, diambil register khusus di mana variabel akan menetap.

Isu-isu code generation



- Algorithm analysis!!
- Latency
 - Memory Latency

Perhatikan juga latency operasi memori, bagaimana agar dua operasi bisa dilakukan secara sequensial. Hati-hati dengan page fault.
 - Branch Latency

Percabangan juga memiliki latency, kompiler harus menghitung keterlambatan yang mungkin terjadi antara percabangan yang menghasilkan instruksi lebih banyak dan yang lebih sedikit.



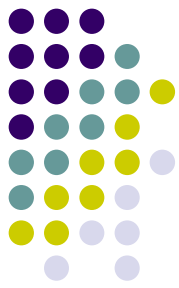
Isu-isu Error Handling

- Jika kompiler menemukan kesalahan atau error, maka kompiler harus menentukan tindakan apa yang harus dilakukan. Kesalahan dapat berupa:
 - **Kesalahan Leksikal**
 - Contoh : salah mengeja/menulis keyword; THEN ditulis TEN
 - **Kesalahan Sintaks**
 - Contoh : salah meletakkan kurung buka atau kurung tutup;
 $jml = x + (a*(c+b)$
 - **Kesalahan Semantik**
 - Tipe data salah: misal sudah dideklarasikan tipe integer, tapi diisi string
 - Variabel belum didefinisikan tapi sudah dioperasikan

Langkah-langkah penanganan kesalahan



- Deteksi kesalahan
- Laporkan kesalahan
- Perbaiki kesalahan selama bisa diperbaiki, atau biarkan user yang membenarkannya.



Pelaporan Kesalahan

- Kode kesalahan
- Pesan kesalahan dalam bahasa natural
- Nama/atribut yang salah
- Tipe data yang terkait

Contoh:

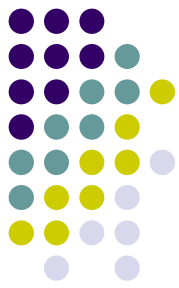
- `error 150: unknown identifier`
- `error in line 10: class not found exception:
java.lang.Sytem.out`

Reaksi Kompiler



- Reaksi yang tidak dapat diterima (tidak melaporkan error):
 - Kompiler crash/hang.
 - Kompiler masih dalam loop yang tidak pernah berhenti.
 - Menghasilkan program obyek yang salah. Kompiler akan terus bekerja menghasilkan kode obyek yang salah, hal ini berbahaya jika tidak diketahui programmer. Kesalahan baru diketahui pada saat runtime

Reaksi Kompiler



- Reaksi yang benar tapi kurang benar
 - Menemukan error, melaporkan error, lalu berhenti. Hal ini muncul jika kompiler menganggap bahwa kemungkinan terjadi error sangat kecil sehingga kemampuan mendeteksi error hanya satu untuk setiap kali kompilasi. Programmer akan membuang waktu untuk melakukan kompilasi untuk tiap error yang ada.
 - Contoh: Program-program Interpreter



Reaksi Kompiler

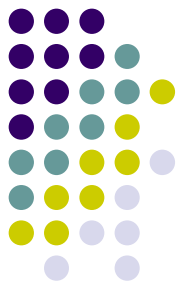
- Kompiler menemukan error, melaporkan dan kemudian melakukan :
 - Recovery (pemulihan) dan menemukan error lain jika masih ada dalam source code itu Contoh: Delphi/Pascal/C++
 - Repair (perbaiki kesalahan), lalu melanjutkan proses translasi dan melanjutkan membuat program obyek yang valid. Contoh?
 - Challenge : kompiler tahu apa maksud programmer dan mengganti kesalahan sesuai dengan maksud programmer itu.

Error Recovery



- Bertujuan mengembalikan kondisi parser ke kondisi stabil, sehingga bisa melanjutkan proses parsing selanjutnya sehingga semua error bisa terbaca.
- Strateginya adalah:
 - Proses Ad Hoc
 - Reaksinya tergantung kompiler, tidak terikat pada aturan tertentu. Disebut juga Special Purpose Error Recovery
 - Syntax Directed Recovery
 - Melakukan recovery berdasarkan sintaks.
 - **Contoh:**
 - Begin
 - A := A + 1
 - B := B + 1;
 - C := C + 1;
 - End
 - Maka notasi BNF nya : begin<statement>?<statement>;<statement>;end
 - Tanda ? akan otomatis diganti menjadi “;”

Error Recovery

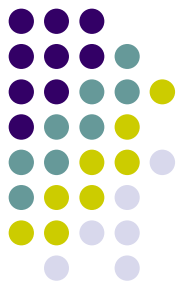


- Secondary Error Recovery
 - Berguna untuk melokalisir error:
 - Panic Mode
 - Maju terus dan mengabaikan teks sampai bertemu “;”
 - Contoh:
IF A = 1
B := TRUE;
 - Unit Deletion
 - Menghapus keseluruhan unit / blok bagian error. Tipe ini tetap memelihara kebenaran sintaks dari program.

Error Recovery



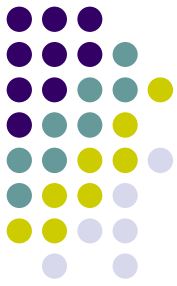
- Context Sensitive Recovery
 - Jika ada variabel yang belum dideklarasikan, maka kompiler akan mengira-ira tipe datanya berdasarkan kemunculannya.



Error Repair

- Bertujuan untuk mengubah source program menjadi benar agar bisa melanjutkan proses pembuatan program akhir.
- Mekanisme Ad Hoc : tergantung kompilernya (terserah)
- Syntax Directed Repair
 - Mengganti / menyisipkan bagian yang error/salah

THE END



- Thanks for studying compiler
- God Bless You on your Final Exam!
- Open books
- Final Test : Bab 5 – Bab 11!