۳ اصول ترجمه زبان (۲)

تعریف و مراحل

ترجمه: تبدیل برنامه‌ای از یک زبان به زبان دیگر

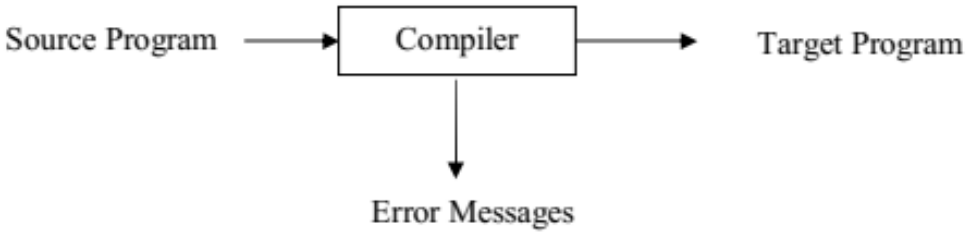
فرآیند ترجمه به‌صورت منطقی:

- تحلیل
  - شامل فازهای تحلیل لغوی، تحلیل نحوی، تحلیل معنایی و تولید کد میانی
- ترکیب
  - شامل فازهای بهینه‌سازی کد و تولید کد نهایی

تعریف و مراحل

کامپایلر

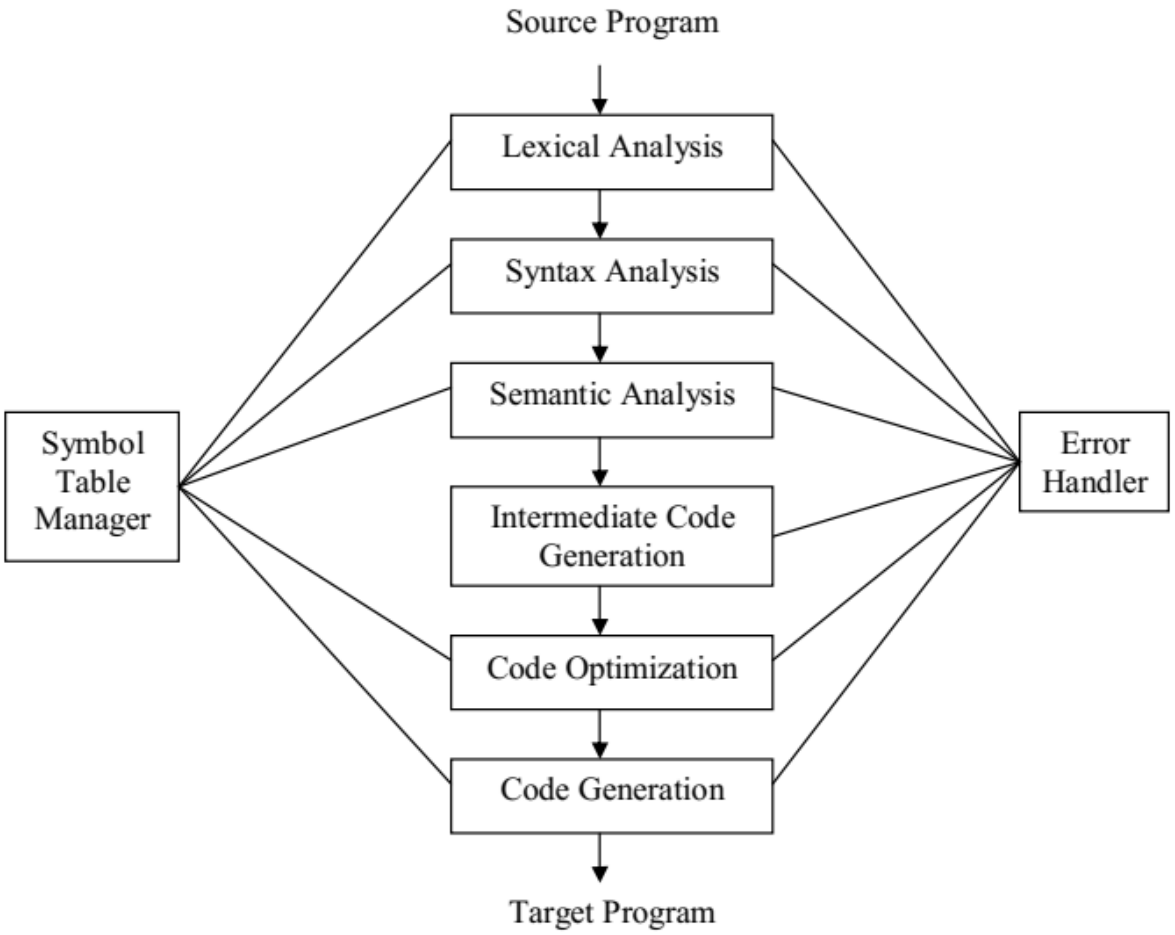
نرم‌افزاری که برنامه نوشته شده در یک زبان را خوانده و از روی گرامر آن ساختار برنامه را به‌دست آورده و آن را به برنامه معادل در زبان دیگر ترجمه می‌کند.



مراحل ترجمه (Compile)

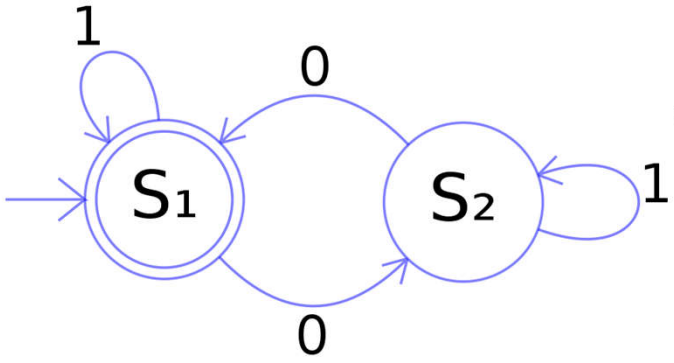
- تحلیل لغوی
- تحلیل نحوی
- تحلیل معنایی
- تولید کد بینابینی
- بهینه‌سازی کد
- تولید کد نهایی

ارتباط بین مراحل ترجمه



مراحل ترجمه – تحلیل لغوی (Lexical Analyzer)

- خواندن برنامه ورودی به‌صورت کاراکتر به کاراکتر و تبدیل آن‌ها به دنباله‌ای از نشانه‌ها(توکن، واحدهای لغوی)
- انواع نشانه‌ها
  - کلمات کلیدی، عملگرها، جداکننده‌ها، ثابت‌ها و شناسه‌ها(اسامی توابع، رویه‌ها، اسامی انتخاب شده توسط کاربر)



- مدل اصلی مورد استفاده برای تحلیل‌گر لغوی، ماشین خودکار متناهی است
- مفهوم تحلیل لغوی ساده، ولی این مرحله بسیار زمان‌بر می‌باشد

مراحل ترجمه – تحلیل نحوی (Syntax Analyzer)

- ایجاد ساختار سلسله مراتبی با استفاده از توکن‌های تولید شده توسط تحلیل لغوی(درخت تجزیه یا Parse Tree)
- نمایش ساختار نحوی توسط درخت‌های تجزیه
- بررسی برنامه از نظر خطاهای نحوی

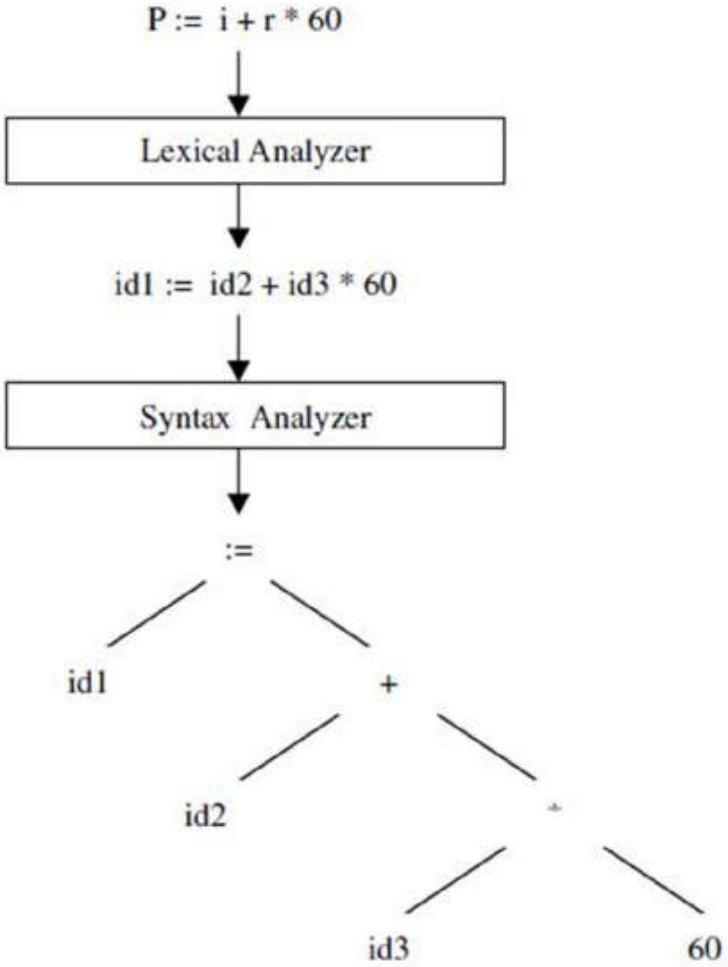
مراحل ترجمه – تحلیل معنایی (Analyzer Semantic)

- تحلیل معنایی مهمترین مرحله ترجمه است
- برنامه ورودی با استفاده از درخت تجزیه، از نظر خطاهای مفهومی مورد بررسی قرار می‌گیرد
- انجام اعمال جانبی نظیر: کشف خطاها، نگهدرای جدول نمادها و اجرای دستورات زمان ترجمه در این مرحله
- یافتن خطاهایی که در زمان تحلیل نحوی مشکل است همانند: خطاهای نوع

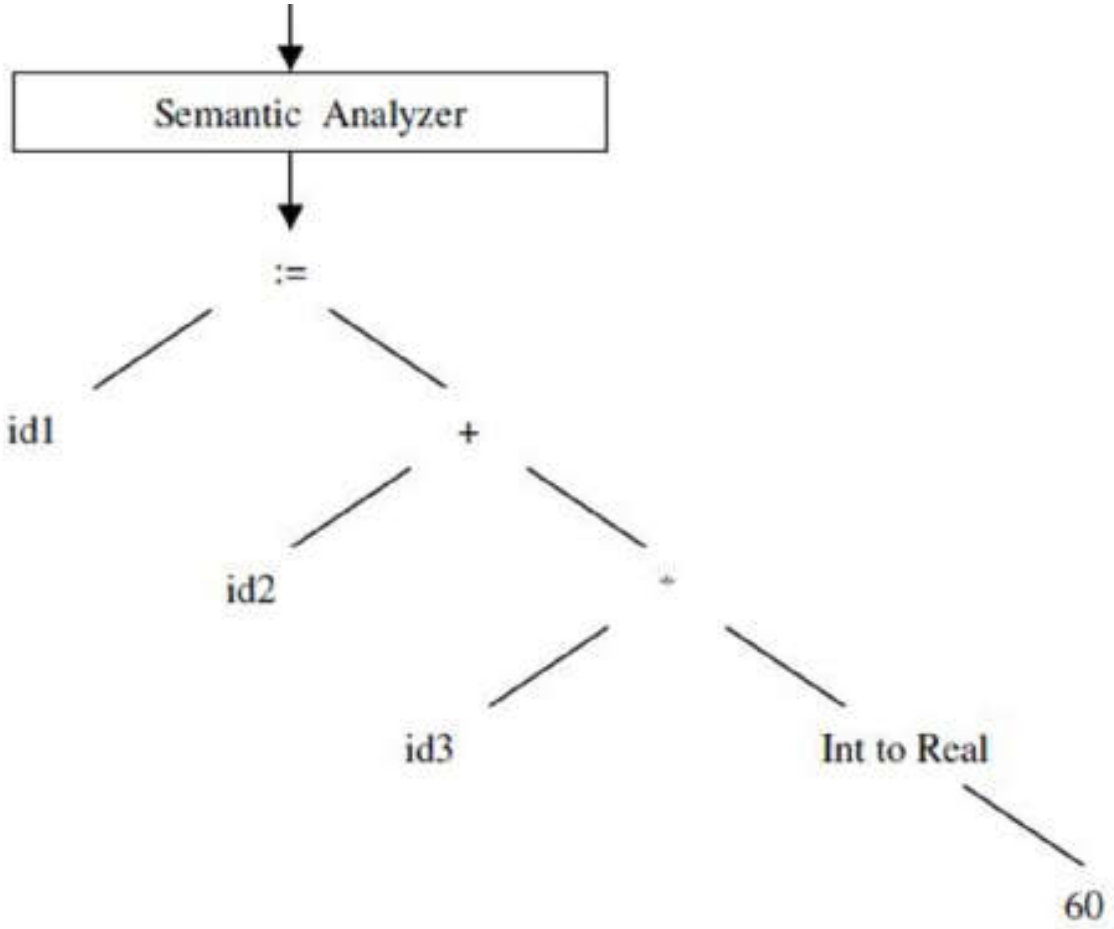
مراحل ترجمه – تولید کد میانی، بهینه‌سازی، تولید کد نهایی

- تولید کد میانی (Intermediate Code Generation)
  - تولید یک برنامه میانی معادل برنامه اصلی، به یک زبان میانی (شبیه به زبان اسمبلی یا خود زبان اسمبلی)
- بهینه‌سازی کد (Code Optimization)
  - سعی در بهبود کد میانی تولید شده
  - هدف: از لحاظ اجرایی سریعتر و مصرف کمتر حافظه
- تولید کد نهایی (Code Generation)
  - تبدیل هر کدام از کدهای بهینه شده به مجموعه‌ای از دستورات زبان ماشین که عملکرد مشابهی دارند

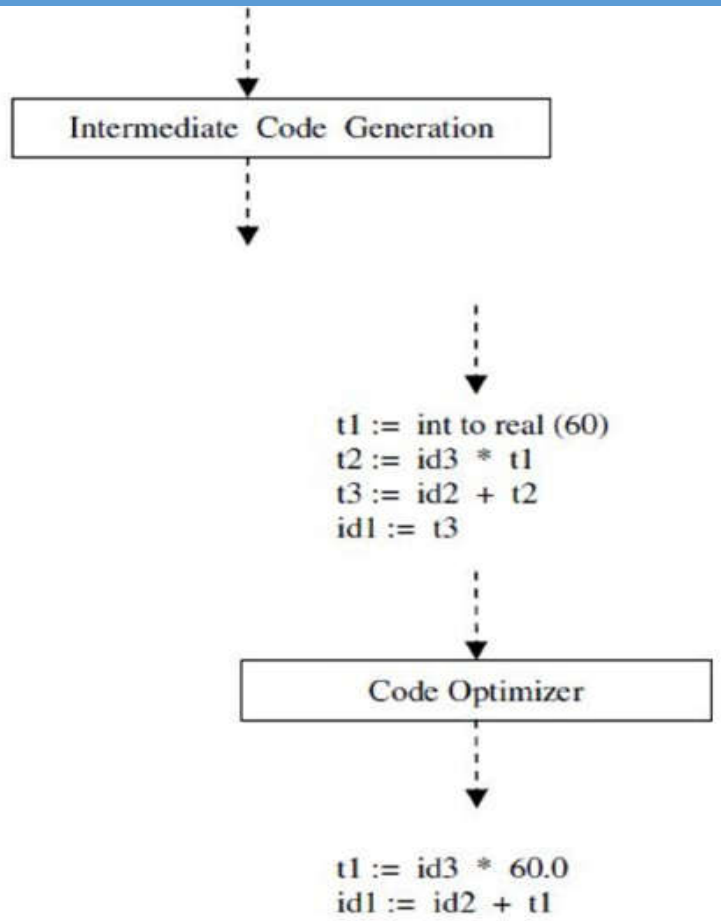
مراحل ترجمه - مثال



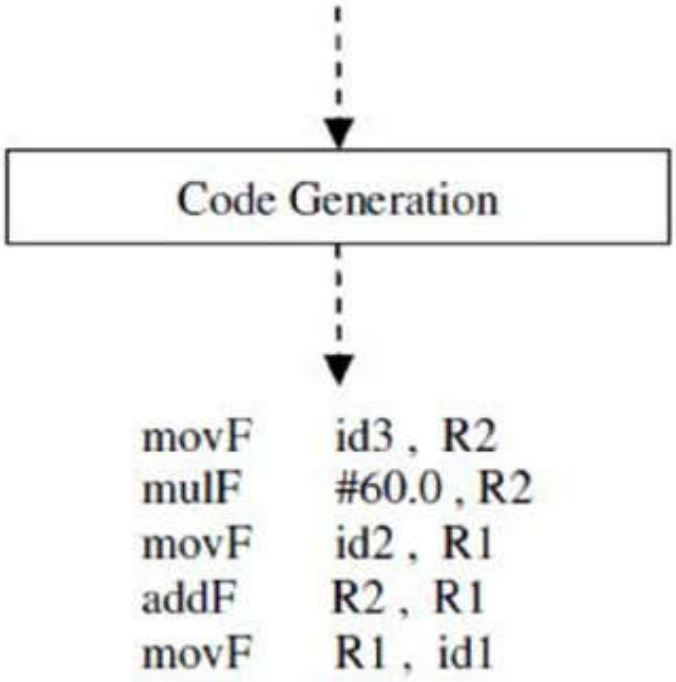
مراحل ترجمه - مثال



مراحل ترجمه - مثال



مراحل ترجمه - مثال



روش‌های تعریف زبان – تشخیص دهنده

- تشخیص دهنده (Recognizer)

- زبان L، الفبای $\sum$ ، دستگاه تشخیص(R)
- همانند فیلتر عمل می‌کنند
- ناکارآمدی و طولانی بودن عمل تشخیص به دلیل نامتناهی بودن اغلب زبان‌ها
- R باید طوری طراحی شود که تشخیص دهد رشته ورودی در زبان L وجود دارد یا خیر
- بخشی از کامپایلر که وظیفه تحلیل نحوی را بر عهده دارد
- تنها رشته‌های کاراکتری موجود در برنامه را بررسی می‌کند

روش‌های تعریف زبان – مولد

- مولد (Generator)
  - راهکار تولید زبان استفاده از گرامر است
    - گرامرها برای توصیف نحو زبان مورد استفاده قرار می‌گیرند
- دسته‌بندی زبان‌ها براساس توصیف گرامر چامسکی
  - زبان طبیعی(Natural Language)
  - زبان حساس به متن (Context Sensitive Language)
  - زبان مستقل از متن (Context Free Language)
  - زبان منظم (Regular Language)

مناسب برای توصیف نحو زبان‌های برنامه‌سازی

توصیف نشانه‌های موجود در زبان با گرامر منظم

توصیف تعداد زیادی از زبان‌های برنامه‌سازی با گرامر مستقل از متن

گرامر BNF

روش نشانه‌گذاری جدیدی برای توصیف نحو زبان و معادل گرامرهای مستقل از متن

عناصر گرامر BNF

- مجموعه‌ای از نمادهای پایانی(Terminal symbols)
  - تکواژه‌هایی از نشان‌های زبان برنامه‌سازی
- مجموعه‌ای از نمادهای غیر پایانی(non Terminal symbols)
  - نمایش دسته‌های مربوط به ساختار نحوی
- مجموعه‌ای از قوانین
  - نمایش چگونگی تولید یک نماد غیر پایانی درون دنباله‌ای از نمادهای غیر پایانی و پایانی

BNF

BNF برای ساختارهای نحوی، از انتزاع استفاده می‌کند

دستور انتساب در جاوا می‌تواند با انتزاع <assign> نمایش داده شود

انتزاع، نماد غیر پایانی نامیده می‌شود

نشان‌ها، نماد پایانی نامیده می‌شود

درخت تجزیه (Parse Tree)

ساختاری سلسله مراتبی برای نمایش جملات زبان

مثال: A = B+C*A