دانشگاه آزاد اسلامی واحد هشترود

کامپایلر ها
اصول، ابزار ها، روش ها

Compilers principles, techniques and Tools

مولف: آلفردوی. آهو – راوی سدی - جفری دی. المن

مدرس: مهندس باقری نیاء

تهیه کننده: سعدی همتی

وبلاگ جزوات: www.cnh86.blogfa.com

بهار – 86

به نام خدا

جلسه اول

ترجمه:

عبارت است از تبدیل یک ساختار از یک زبان(زبان مبدا) به زبان دیگر(مقصد).

مترجم:

برنامه ای است که عمل ترجمه از زبان مبدا به زبان مقصد را انجام می دهد.

انواع مترجم:

☐ **کامپایلر:** اگر زبان مبدا یکی از زبان های سطح بالا(پاسکال،فرترن PL/1) و زبان مقصد زبان ماشین باشد به چنین مترجمی کامپایلر گویند.

☐ **مفسر:** مفسر نیز یک زبان سطح بالا را دستور به دستور ترجمه کرده و اجرا می کند و زبان ماشین تولید نمی کند.

نکته۱: فرق کامپایلر با مفسر در این است که کامپایلر کد قابل اجرا تولید می کند ولی برای اجرای یک برنامه توسط مفسر هر بار می بایست ترجمه و اجرا صورت گیرد..

نکته۲: حجم مفسر از حجم کامپایلر کمتر است ولی سرعت کامپایلر بیشتر است

نکته۳: از لحاظ مصرف حافظه بهتر است از مفسر استفاده شود، چون مفسر فضای کمتری اشغال می کند و از لحاظ امنیت کامپایلر بهتر است زیرا کل برنامه را یک جا ترجمه می کند..

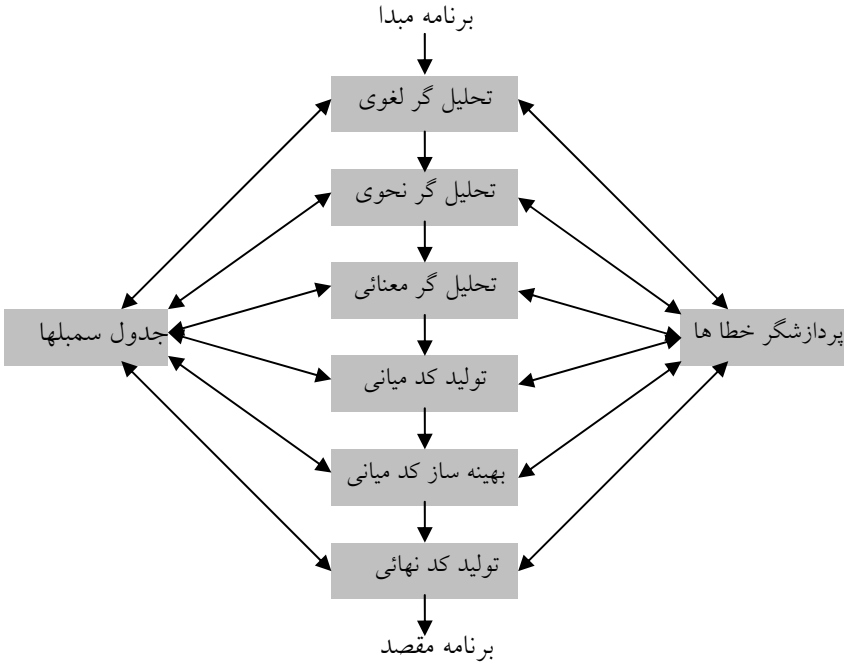
☐ **اسمبلر:** ورودی اسمبلر زبان اسمبلی(کد های یادمان) می باشد که آن را به زبان ماشین تبدیل می کند. اسمبلر های اولیه هر دستور زبان اسمبلی را تبدیل به یک دستور زبان ماشین می کردند که جهت سرعت بخشیدن به عمل ترجمه ، اسمبلر هائی ابداع گردید که قادرند یک دستور اسمبلی(دستور کلان) را تبدیل به چندین دستور زبان ماشین کنند.

☐ **پیش پردازنده:** یک زبان سطح بالا را تبدیل به یک زبان سطح بالای دیگری می کند که اغلب زبان مبدا شکل توسعه یافته زبان است مثل Turbu C، و زبان مقصد شکل استاندارد زبان است مثل ANSI C

مراحل کامپایل در شش مرحله صورت می گیرد.

۱- تحلیل لغوی(Lexical analysis)
۲- تحلیل نحوی(syntax analysis)
۳- تحلیل معنائی(semantic analysis)
۴- تولید کد میانی(intermediate code generation)
۵- بهینه سازی کد(code optimization)
۶- تولید کد نهائی(code generation)

این شش بخش به همراه بخش های "پردازش خطا" و "جدول سمبل ها" یک کامپایلر را تشکیل می دهند ارتباط بین این بخش ها در زیر نشان داده شده است.



تحلیلگر لغوی:

در یک کامپایلر به تحلیل خطی(Linear Analysis) ، تحلیل لغوی(lexical Analysis) ، تحلیل پویش(scaning Analysis) گفته می شود. کاراکتر ها را از چپ به راست خوانده و با رسیدن به یک جدا کننده آنها را در گروه های منطقا به هم مرتبط به نام Token قرار می دهد و Token ها که خروجی تحلیل گر لغوی می باشند در جدول سمبل ها با فرمت خاصی ذخیره می شوند و همچنین به عنوان ورودی تحلیل گر نحوی استفاده می شوند به عبارتی تحلیل گر لغوی واسط بین برنامه مبدا و تحلیلگر نحوی است.

انواع مختلف نشانه ها(Tokenها) عبارتند از

کلمات کلیدی(keywords)

عملگر ها(operators)

ثابت ها(literals)

شناسه ها(identifiers) که به اسامی متغیر ها، توابع، رویه ها و به طور کلی اسامی که کاربر انتخاب می کند گفته می شود.

جداکننده ها(delimiters) { *space*, *tab*, ↵, ;, : }

ساختار Token برای a=b+2

Token type	Token value
id	a

نکته:

تحلیلگر لغوی comment ها(توضیحات) را رد می کند و فاصله های خالی(white space) را که کاراکتر های شناسه ها را که از هم جدا می کند را حذف می کند و اگر خطائی رخ بدهد آن را گزارش می دهد. مثلا با دیدن عبارت " 6a " تحلیل گر لغوی خطائی را گزارش می کند.

تحلیلگر نحوی:

تحلیل سلسله مراتبی(Hierachical analysis) ، تجزیه(parsing analysis) یا نحوی(syntax analysis) نامیده می شود. تحلیلگر نحوی با داشتن گرامر مربوط به زبان و Token های دریافتی از تحلیلگر لغوی برای یک دستور درخت اشتقاق آن را ساخته و اگر دستور مربوط به زبان نباشد خطا گزارش می دهد وگرنه درخت اشتقاق را به عنوان خروجی به تحلیلگر معنائی می دهد.. مثلا در زبان C اگر دستوری به شکل $for(i=0;i++)$ داشته باشیم خطای نحوی رخ میدهد.

تحلیلگر معنائی:

مهمترین وظیفه تحلیلگر معنائی کنترل نوع یا(Type cheking) است. اگر تبدیل نوع مجاز باشد عمل تبدیل نوع را انجام می دهد وگرنه خطای تبدیل نوع را گزارش میدهد

تبدیل نوع $\leftarrow float + int$

خطا $\leftarrow float + char$

تولید کننده کد میانی:

در این بخش برنامه ورودی به یک زبان میانی تبدیل می شود. میتوان این زبان میانی را به برنامه ای برای یک ماشین انتزاعی تشبیه کرد. این زبان میانی حداقل بایستی خواص ذیل را دارا باشد.

سهولت تولید کد : زبان ماشین منطقی تا حد امکان باید ساده در نظر گرفته شود تا تولید کد برای آن آسان باشد.

سهولت ترجمه به زبان مقصد: زبان ماشین منطقی باید به گونه ای باشد که تبدیل آن به زبان ماشین آسان باشد.

بهینه سازی کد میانی:

در این بخش سعی می شود تا کد میانی به صورتی در آید که سریع تر اجرا شود مثلا در کد a=func(b) +func(b) به جای دو بار فراخوانی تابع func() این تابع با پارامتر b یک بار فراخوانی شده و مقدار برگشتی آن در یک متغیر موقت ریخته می شود و سپس از مقدار متغیر موقت استفاده می شود.

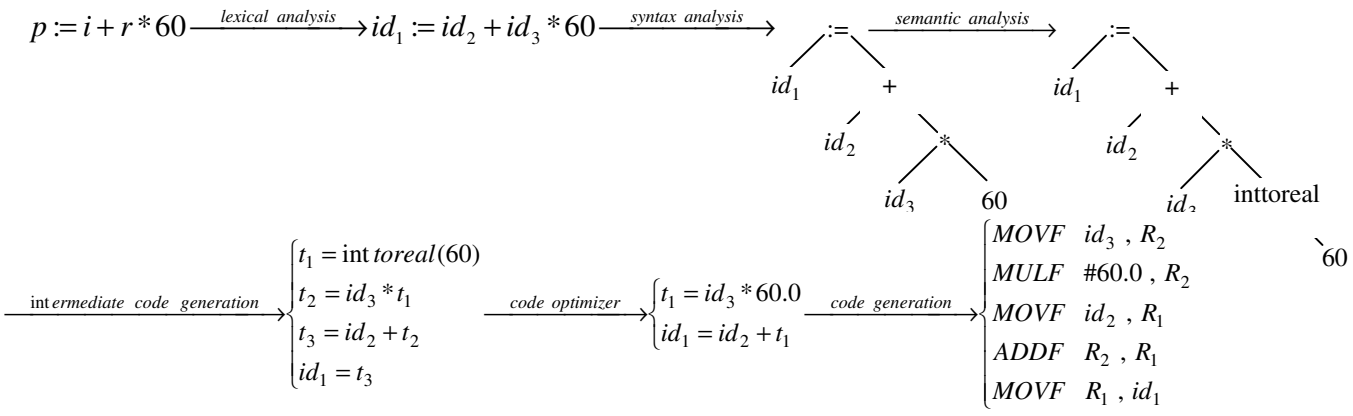
دلیل این که عمل بهینه سازی همراه با کد میانی انجام نمی شود چیست؟

جواب- عمل تولید کد میانی همراه با درخت انجام می شود و کار کردن با درخت مشکل است ولی بهینه ساز کد، یک فایل را به عنوان ورودی می گیرد.

تولید کننده کد نهائی:

در این مرحله کد اسمبلی قابل اجرا بر روی ماشین ایجاد می شود. در واقع کد تهیه شده میانی که در آن از ثبات استفاده نشده می بایست به کدی تبدیل شود که در آن از ثبات استفاده می شود.

به عنوان مثال شکل زیر مراحل کامپایلر p=i+r*60 و تاثیر هر مرحله از کامپایل بر روی این دستور را نشان می دهد(فرض شده است که i,p,r همگی از نوع real هستند)



نکات تکمیلی جلسه اول

- بخش front-end یک کامپایلر برای تمام ماشین ها یکسان است ولی back-end برای ماشین های مختلف متفاوت است
- اگر کاراکتر غیر مجاز در متن برنامه نوشته شود تحلیل لغوی(scanner) آن را تشخیص میدهد(علوم کامپیوتر-۷۹)
- در فرایند کامپایل یک برنامه توکن ها در مرحله آنالیز لغوی(Lexical) شناسائی میشوند.(مهندسی کامپیوتر آزاد-۷۱)
- در مرحله تحلیل لغوی کلیه شناسه های موجود در برنامه وارد جدول علائم(symbol Table)میشود.(مولف-راهیان ارشد)
- back-end بخشی از مرحله بهینه سازی و مرحله تولید کد نهائی که وابسته به ماشین است، می باشد.(مولف- راهیان ارشد)
- تفاوت بین compiler و preprocessor در این است که کامپایلر تحلیل لغوی و معنائی را روی برنامه انجام میدهد، در حالی که preprocessor این کار را انجام نمیدهد(مولف- راهیان ارشد)
- دلیل استفاده ازfront-end و back-end در کامپایلر ها، تدارک ترکیب چند front-end و back-end در یک خانوادهٔ کامپایلر است(مولف-راهیان ارشد)
- در صورتی که شناسه قبلا اعلان نشده باشد، یک خطای معنائی رخ میدهد.(نوپردازان-مولف)
- کشف خطا های مربوط به ساختار تک تک لغات وظیفه تحلیل گر لغوی است(کتاب-پیام نور)
- در دستور $\text{var } 7temp := integer;$ که به زبان پاسکال می باشد، تحلیل گر لغوی لغت $7temp$ را به عنوان خطا گزارش میدهد زیرا در پاسکال یک شناسه نباید با عدد شروع شود.
- در زبان پاسکال عبارت، $A \quad B :=$ به دلیل عدم رعایت ترتیب صحیح انتساب دارای خطای نحوی می باشد.
- در زبان پاسکال عبارت، $if\,(a = b \quad then$ به دلیل عدم توازن پرانتزها دارای خطای نحوی می باشد.
- تحلیل گر معنائی معنی دار بودن عباراتی که از نظر نحوی درست بوده اند را مورد بررسی قرار میدهد.

دو مزیت عمده کد میانی عبارت است از:

- کامپایلر را می توان مستقل از ماشین نوشت لذا با تغییر ماشین ها و تکنولوژی انها کافی است تنها بخش مولد کد تغییر کند و نه دیگر بخش های کامپایلر.
- یک بهینه ساز کد مستقل از ماشین را می توان برای سریع تر کردن کد میانی استفاده کرد، در نتیجه بدون توجه به نوع ماشین، کد تا حد ممکن بهینه خواهد بود.

به طور کلی کد های میانی شامل انواع زیر می باشد :

۱- کد شبه اسمبلی
۲- درخت خلاصه نحوی
۳- جملات سه آدرسه

☐ با یک پیمایش پسوندی به سادگی می توان از درخت خلاصه نحوی کد اسمبلی تولید کرد.

وظایف تحلیل گر نحوی:

۱- بررسی صحت و درستی ترتیب لغات برنامه(بررسی ساختار برنامه مبدا)

۲- بررسی جریان کنترل(Flow of control checks)،مثلا عبارتBreak در زبان c موجب خروج از نزدیکترین حلقه(switch، for، while) می گردد که در صورت نبودن چنین حلقه ای خطا رخ میدهد.

۳- کنترل منحصر به فردبودن، مثلا نام یک Lable می بایست دقیقا یک بار در برنامه ظاهر شود.

۴- چک کردن نام های مرتبط، مثلا در زبان Ada یک حلقه دارای یک نام در ابتدا و انتهای ساختار خود است که می بایست یکسان باشند.

۵- چک کردن ساختار های تودرتو

برخی از مواردی که توسط تحلیل گر معنائی انجام میشود عبارتند از:

بررسی هماهنگی پارامترها : فراخوانی یک روال باید با تعریف روال هماهنگی داشته باشد. مثلا در زبان C تعداد پارمتر های ارسالی، نوع آنها و ترتیب آنها باید در تعریف تابع هماهنگی داشته باشد.

بررسی و کنترل نوع : در این قسمت نوع عملوند های یک عملگر مورد بررسی قرار می گیرد. مثلا در $a[1.5]:=12$ خطای معنائی وجود دارد زیرا اندیس یک آرایه(a) نبایستی یک عد د اعشاری باشد.

تبدیل نوع : عملیات جمع برای اعداد صحیح و اعشاری متفاوت است برای انجام عمل جمع در عبارت $c = a + b$ در صورتی که c و b از نوع float و a از نوع int باشد، تحلیلگر معنائی عمل ارتقاء نوع را انجام میدهد. بدین معنا که موقتا متغیر a به نوع float تبدیل میشود.

تعریف دوباره متغیر: تحلیلگر معنائی بررسی می کند تا هیچ متغیری دو بار تعریف نشده باشد