کامپایلر
مقدمات

محسن هوشمند
دانشکده تکنولوژی اطلاعات و علم رایانه
دانشگاه تحصیلات تکمیلی علوم پایه زنجان

مثال

x= a+b*60; /* #####*/

Lex

تحلیل لغوی

الگو

l(l+d)*

<id,1><=><id,2><+><id,3><*><num,4>

Yacc

1		...
2	a	...
3	b	...
4	60	

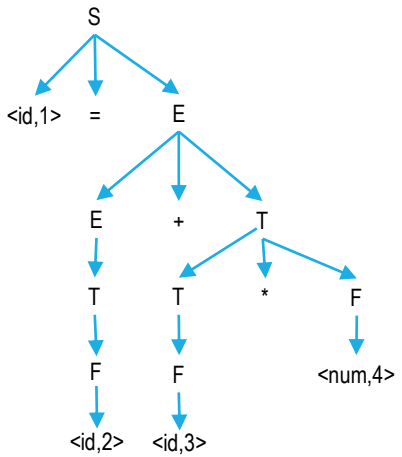
تحلیل نحوی

$S \rightarrow id = E$

$E \rightarrow E + T|E$

$T \rightarrow T * F|F$

$F \rightarrow num$



تحلیل معنائی

درخت تجزیه (معنائی)

م ک م

t_1=int2float(60)
t_2=id_3*t_1
T_3 =id_2+t_2
id1=t_3

بهینه‌سازی کد

t_1=id_3*60.0
id1=id_2+t_1

مولد کد

ld R_2, id_3
mul R_2, R_2, #60.0
ld R_1 , id_2
add R_1, R_1, R_2
mov id_1, R_2

پسا

تحلیل لغوی

پویشگر، اسکنر، lexical analyzer،

اولین مرحلهٔ کامپایلر

تنها بخش با دسترسی مستقیم به کد

- دریافت از پیش‌پرداز
- کد به همراه دخیل کردن ماکروها و فایل‌ها

خواندن رشته نویسه‌های تشکیل دهندهٔ برنامه مبدا

گروه‌بندی نویسه‌ها به دنباله‌های معنادار

- لغت یا واژه
- به دیگر سخن، تبدیل برنامه به قالبی موجز و یکنواخت از تکه‌ها
- تشخیص نوع لغت
- استفاده از عبارت منظم
- ارسال به تحلیل‌گر نحوی در صورت درست بودن املائی یا ارسال پیام خطای املائی

تحلیل لغوی - *ادامه*

تولید تکه به ازای هر لغت <نام تکه و مقدار تکه>

- توکن یا تکه ؟
- نام: علامت مورد استفادهٔ تحلیل‌گر نحو
- مقدار: اشاره به مدخلی در جدول علامت
  - اطلاع مدخل جدول علامت جهت تحلیل معنائی و تولید کد

حذف اطلاعات غیرلازم

افزودن اطلاعات لازم به جدول علامت

تحلیل لغوی - ادامه

لغات x= a+b*60; /* #####*/

۱- x لغتی که که به تکهٔ <id,1> نگاشت می‌یابد
- Id علامت معادل شناسه identifier
- ۱ اشاره به نشانی مدخل متناظر در جدول علامت
- حفظ نام و مقدار در جدول

۲- علامت تخصیص =، لغت نگاشت شده به تکهٔ <=>
- جزء دوم را لازم ندارد

۳- a لغت نگاشتی به <id,2>

۴- + نگاشت به تکهٔ <+>

۵- b نگاشت به <id,3>

۶- * نگاشت به تکهٔ <*>

۷- 60 نگاشت به تکهٔ <num,4>

تحلیل نحو

تجزیه‌گر، پارسر

تشخیص صورت ظاهری برنامه، و بررسی صحبت و درستی ترتیب لغات برنامهٔ مبدأ

استفاده از تکه‌های تولیدی تحلیل لغوی

ایجاد ساختار درختی (درخت تجزیه) از خروجی تحلیل‌گر لغوی

- نمایش میانی

تحلیل نحو

مثال جهت تقریب ذهن

- تحلیل‌گر لغوی
  - سابون شوینده است
  - صابون شوینده است
- تحلیل‌گر نحوی
  - شوینده صابون است

```
for(int i; i<10; i++
{
  x+y=z;
}
```

لغوی درست

نحوی غلط

تحلیل نحو - *ادامه*

پیش‌بینی تکه‌ها با ساختار دستوری

- مشخصات صوری نحو چون دستور مستقل‌ازمتن

خواندن تکه‌ها

- گروه‌بندی آنها به عبارات
- بر اساس مشخصات نحوی

نمایش معمول: درخت نحو

- رأس داخلی: عملیات و فرزندان عبارات (ارگومان‌های) عملیات

ترتیب عملیات متناسب با قرارداد معمول ریاضی

کامپایل گری

ابزاری جهت ترجمه نرم‌افزاری از زبانی به زبان دیگر

نیاز به فهم

- صورت (فرم) form یا نحو (جمله‌شناسی) syntax
- محتوا content یا معنا Meaning
- نیاز به شمائی جهت تبدیل محتوا از زبان مبدا به زبان مقصد

نحو (ساخت) و معنا

تعریف کامل زبان دارای نحو و معنا

نحو

- معمولا دستورهای مستقل‌ازمتن (دم‌ام)
- تعریف دنبالهٔ معتبر کلمات
- مستقل از معنای علائم
- عدم‌امکان توضیح همهٔ جوانب با نحو (سازگاری نوع و قوانین حوزه)

معنا

- جهت پر کردن محدودیت‌های دم‌ام
- به دو وجه ایستا و پویا

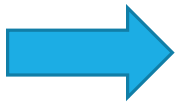
معنای زمان‌اجراء

- طبیعی
- ارزآغازه
- معنای مدلولی

نحو (ساخت) و معنا – *ادامه*

معنای ایستا

- قوانین مبین اعتبار دستورات خوش‌نحو
  - سازگاری نوع عمل‌گر و عمل‌وند
  - تعداد ارگومان‌های درست تابع
- بدون توجه به شکل و فرم، یا صوری
- مثال دستورهای صفت کنوث

$E \leftarrow E + T$ 

$E_r \leftarrow E_{v1} + T_{v2}$
$if\ v1.type = numeric\ and\ v2.type = numeric$
$\quad r.type \leftarrow numeric$
else call error()

- یا درخت نحو انتزاعی

نحو (ساخت) و معنا – *ادامه*

معنای زمان‌اجراء

- طبیعی
  - در صورت درست بودن ارزیابی‌های ایجاد ساختاری، آن‌گاه ساختار نیز درست
  - ML
- ارزآغازه
  - انتزاعی‌تر از مدل‌های عملگری
  - مبتنی بر روابط و مسندها و محمولات

$y > 3$ پس از $y = x + 1$
درست اگر $x + 1 > 3$
$Y=21$ پس از اجرای $x=1$
درست اگر قبل از اجرای $x=1$، Y برابر ۲۱
درصورت نام دیگر Y بودن گزارهٔ قبلی غلط

- معنای مدلولی
  - معنای ریاضیاتی
  - موجز

تحلیل معنا

بهره از درخت نحو و اطلاعات مضبوط در جدول علامت

جهت بررسی سازگاری معنائی برنامه مقصد با تعریف زبان

- تعیین صحت مفهوم جملات
- سنجش درخت تجزیه و تولید «درخت تجزیهٔ سنجیده»

مثال- صابون شوینده را خورد

- لغوی
- نحوی
- معنائی

تحلیل معنا - *ادامه*

اعلان متغیر

- مثال

```
int x;
y = x + 2;
```

وارسی نوع عملوندها

- مثال- اندیس ارایه عدد طبیعی نه عدد اعشاری
- مثال

```
int x;
float y, z;
z = x + y;
```

- مثال

```
int x;
char ch;
x = x + ch;
```

تحلیل معنا - *ادامه*

وارسی هماهنگی پارامترها

- مثال

```
int f(int x, int y)
{
        return x+y;
}
int main()
{
        int a = 1;
        int b = 3;
        float c = 1.5;
        a = f(b,c);
        a= f(b,b,b);
}
```

مولد کد میانی

امکان تولید یک یا چند نمایش میانی

درخت نحو
- معمولا جهت تحلیل معنائی و نحوی

تولید نمایش میانی شبه‌ماشین یا سطح پائین
- به مثابه برنامه‌ای برای ماشین مجازی

هر کامپایلر کد میانی خاص خود

ویژگی‌ها
- سادگی تولید
- سادگی تبدیل به زبان ماشین

مثال کدهای سه نشانی
- حداکثر یک عملگر
- نام موقت برای مقادیر میانی
- گاهی اوقات کمتر از سه عملوند

x = y + z; => add x, y, z

بهینه‌ساز کد میانی

تا مرحله قبل یکسان برای تمامی بسترها

یا مستقل از ماشین یا وابسته به ماشین

کد مقصد بهتر

- اعمال تغییرات در برنامه بدون تغییر در عملکرد ان
- ضمانتی وجود ندارد که تولیدی کامپایلری سریعتر از هر کد دیگری باشد
- امروزه دارای اهمیت بیشتر

امکان وجود پیچیدگی یا افزونگی

شامل چند زیرمرحله

؟

- سریع‌تر
- کوتاه‌تر
- کم‌مصرف‌تر

مثال

- حذف صفر در جمع یا یک در ضرب، امثالهم

نظریه تصمیم

بهینه‌ساز کد میانی - اهداف

درست باشد
- حفظ معنای برنامه ترجمه

بهبود کارکرد بسیاری از برنامه‌ها

دارای زمان بهینه‌سازی منطقی

مدیریت‌پذیر بودن مهندسی موردنیاز

استفاده از نظر و آزمایش

مولد کد

ورودی: نمایش میانی

ترجمهٔ کد میانی بهینه به زبان اسمبلی
- خروجی وابسته به نوع اسمبل‌گر

اسمبل‌گر: ترجمه خروجی مولد کد به زبان ماشین

اگر زبان مقصد کد مطلق ماشین
- مشخص کردن محل ثبات‌ها و حافظه
- ...

gcc
- تغییر خودکار مقصد retarget automatic construction
  - امکان تولید کد مربوط به سی نوع معماری کامپیوتر مختلف چون Intel و Sparc و PowerPC
  - داری حداقل شش پیشا (سی و سی++ و فرترن و آیدا و جاوا)

بهینه‌ساز کد وابسته به ماشین

ثبات

دستورهای سریعتر

روش‌های نشانی‌دهی

اجتناب از ارجاع بی‌مورد به حافظه

استفاده از عملیات ساده‌تر

کامپایل‌گر، اسمبل‌گر، مفسر

کامپایل

- خواندن کل برنامه نوشته شده در زبان سطح بالا
- ترجمه به «زبان ماشین» متناظر
- بی‌خطا در صورت انجام ترجمه
  - در غیر این‌صورت اعلام خطا با خط متناظر خطای رخ‌داده

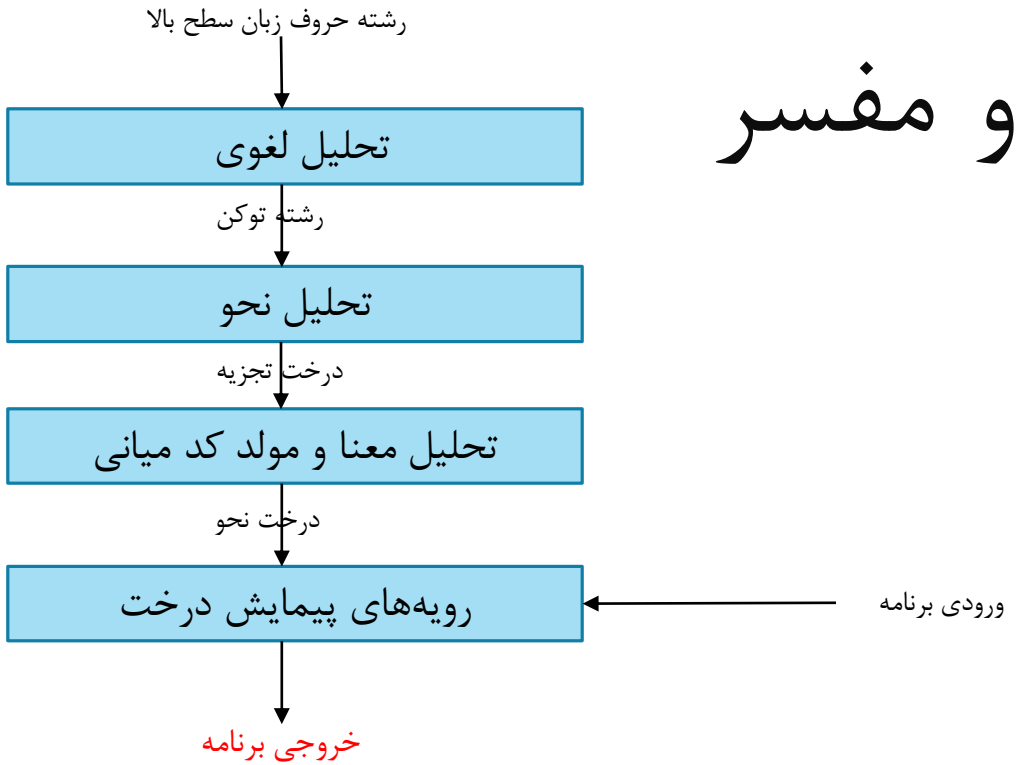
اسمبل

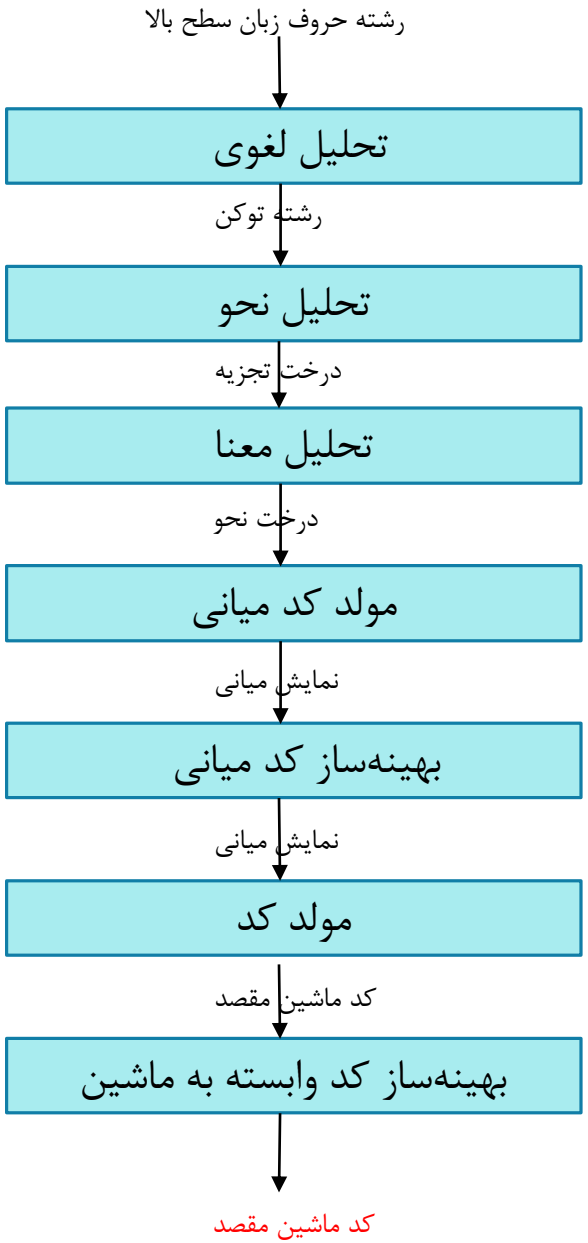
- ترجمه برنامه نوشته شده با زبان اسمبلی به زبان ماشین

مفسر

- ترجمهٔ خط به خط
- اجرای هر خط قبل از رفتن به خط بعدی
- در صورت وجود خطا در خط، توقف و پیام خطا

کامپایل‌گر و مفسر





کامپایل‌گر و مفسر – مزایا و معایب

مفسر

مزایا

- سهولت در اشکال‌زدائی
- انعطاف‌پذیری بالا
- پیاده‌سازی آسان
- حمل‌پذیری

معایب

- نیاز به تفسیر دوباره
- سرعت پائین اجرا
- نیاز به همراه بودن همیشگی مفسر جهت اجرای کد
- دسترسی به کد مبدا

کامپایل‌گر

مزایا

- ایجاد کد شی (ابژهٔ) مستقل
- اجرای کل برنامه
- عدم نیاز به کامپایل مجدد
- عدم نیاز به کد مبدا
- سرعت بالای اجرا بعد از کامپایل

معایب

- اشکال‌زدائی طولانی‌تر
- حمل‌پذیری کمتر

جدول علامت

ساختمان داده تولیدی کامپایلر

- جهت یافتن و ردگیری معنای متغیرها
- ذخیره اطلاع حوزه و قید متغیرها

تولید در مراحل تحلیل لغوی و نحو

جدول علامت - *ادامه*

در مراحل مختلف

- تحلیل لغوی
  - ایجاد مدخل‌های جدول چون تکه‌ها
- تحلیل نحوی
  - افزودن اطلاع چون نوع، حوزه، ابعاد، مرجع،
- تحلیل معنائی
  - استفاده از اطلاع مدخل‌ها جهت بررسی معنا
- تولید کد میانی
  - نحوه تخصیص حافظه حین اجرا
- بهینه‌سازی کد
  - استفاده جهت بهینه‌سازی وابسته به ماشین
- تولید کد مقصد
  - تولید کد با استفاده از نشانی شناسه‌های در جدول

جدول علامت - *ادامه*

اطلاع ذخیره شده:

- نام متغیرها و ثابت‌ها
- نام توابع و رویه‌ها
- رشته‌ها
- مقادیر موقت

اطلاعات موردارجاع از جدول

- نوع داده و نام
- رویه‌های اعلان
- ارجاع با مقدار یا اشاره
- تعداد و نوع ارگومان‌های تابع
- ...

روش‌های پیاده‌سازی: فهرست، فهرست پیوندی، جدول هش، درخت جستجو دودوئی

جدول علامت- مثال

```
/ Define a global function
int add(int a, int b)
{
        int sum = 0;
        sum = a + b;
        return sum;
}
```

NAME	TYPE	SCOPE
add	function	global
a	int	function parameter
b	int	function parameter
sum	int	local

ابزارهای تولید تجزیه‌گر

ابزارهائی جهت پیاده‌سازی مراحل مختلف کامپایلر

- کل کامپایلر (کامپایلر کامپایلرها) یا بخش‌هایی از آن
- معمولا درای بخش‌های پویشگر و تجزیه‌گر

مولد پویشگر

- تولید تحلیل‌گر لغت
  - از ورودی شامل توضیح عبارت منظم
- Lex

مولد تجزیه‌گر

- تولید تجزیه‌گر
  - با داشتن توضیح دستور زبان برنامه‌نویسی
  - یا با دستور مستقل از متن
  - مفید به دلیل پیچیدگی بالای این مرحله
  - مثال: yacc و Pic و EQM

ابزارهای تولید تجزیه‌گر - *ادامه*

بعضی دارای مدیر جدول علامت، ارزیاب دستور، تولید کد

بسته‌های پیشرفته‌تر: تصحیح خطا

موتورهای ترجمهٔ مستقیم نحو

- ایجاد کد میانی سه نشانی از درخت تجزیه
- پیمایش درخت تجزیه

مولد کد خودکار

- تولید زبان ماشین
- استفاده از قوانینی جهت ترجمهٔ هر عمل زبان میانی
- تطبیق الگو

ابزارهای تولید تجزیه‌گر - *ادامه*

راحت‌تر

- یادگیری و خواندن و فهم آسان‌تر
- دارای کامپایلرهای با کیفیت
- غالبا تولید کد بهتر
- رخ دادن کمتر خطای کامپایل
- کامپایلر ارزان‌تر، سریعتر، مطمئن‌تر، پراستفاده‌تر
- پیام‌های تشخیص و ابزارهای توسعه بهتر

تشخیص خطا و بازیابی

فرایند مدیریت خطا: یافتن و گزارش

دارای توابع

- تشخیص
- گزارش
- بازیابی

انواع خطا

- زمان کامپایل
  - خطای لغوی
  - خطای نحوی
  - خطای معنایی
- زمان اجراء
  - خطای لغوی
  - طول غیرمجاز
  - نویسه بی‌مورد
  - رشته غلط
  - خطای نحوی
  - خطا در ساختار
  - عمل غلط
  - کلیدواژه غلط
  - پرانتز نامتوازن
  - خطای معنایی
  - ناسازگاری عمل‌وندها
  - متغیر اعلان نشده
  - عدم رعایت آرگومان‌ها

تشخیص خطا و بازیابی - ادامه

بازیابی

- زمان کامپایل
  - خطای لغوی
  - بازیابی حالت هراس (زدودن هراس)
    - غصه خوردی؟!
    - زدودن غلط
  - خطای نحوی
    - زدودن هراس
    - تصحیح جمله
    - تولید خطا
    - تصحیح سراسری
  - خطای معنایی
    - تبدیل نوع در صورت عمل روی عمل‌وندهای هم‌نوع
    - شناسه تعریف نشده: تعریف آن و افزودن به جدول علامت

منابع

[کوپر]

[اژدرها]

[فیشر]

[اسکات]

“Compiler Design Tutorial: What is, Types, Tools, Example,” https://www.guru99.com/compiler-design-tutorial.html

“C++ Program to implement Symbol Table,” https://www.geeksforgeeks.org/cpp-program-to-implement-symbol-table/?ref=rp

“Functional, Declarative, and Imperative Programming,” https://stackoverflow.com/questions/602444/functional-declarative-and-imperative-programming