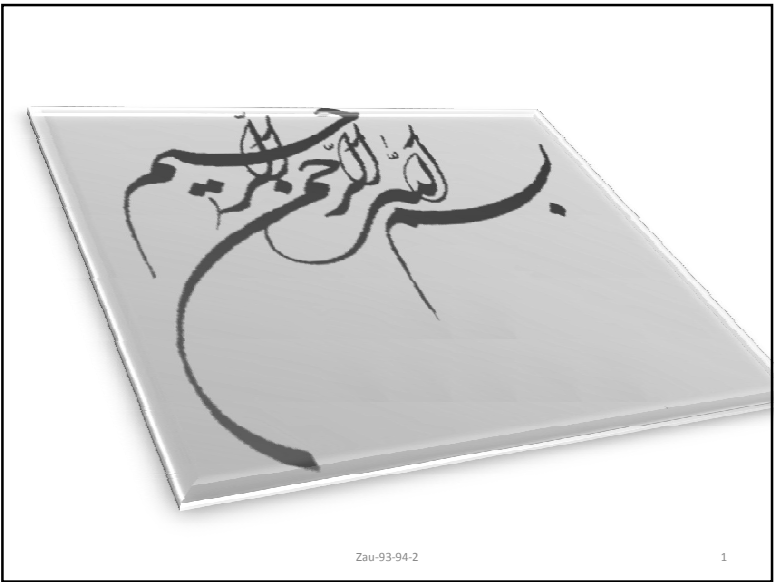
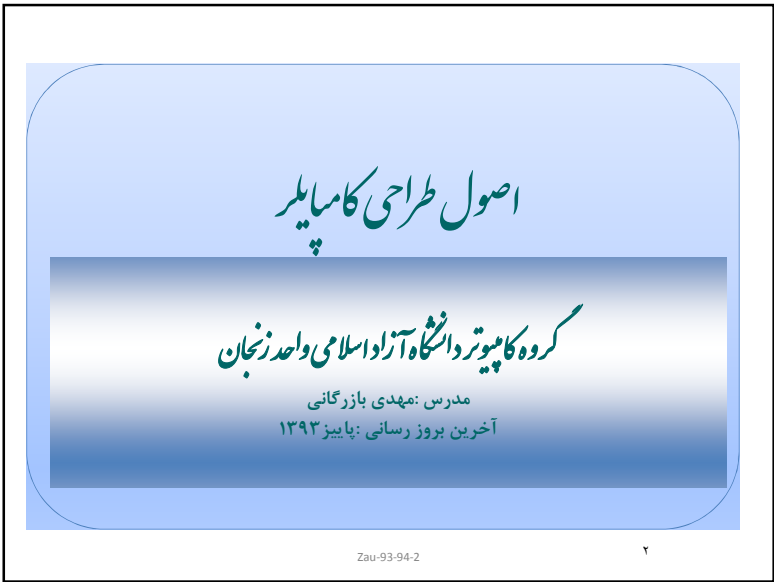




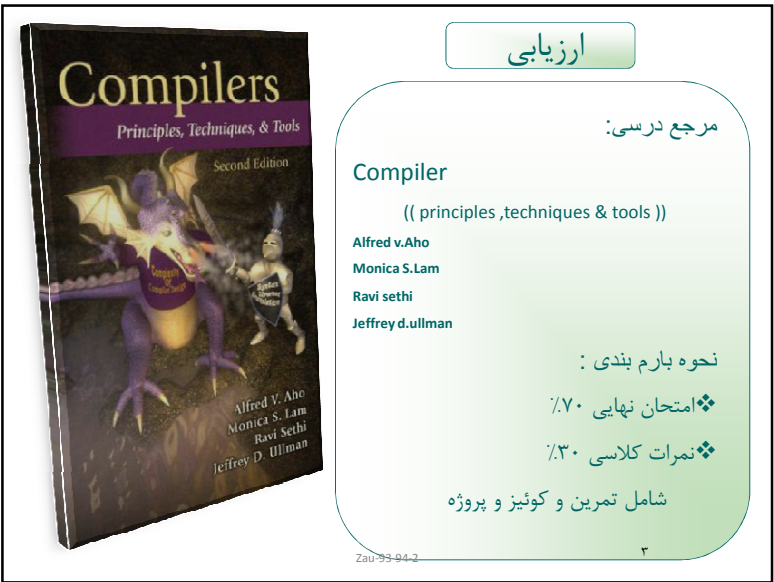
Zau-93-94-2

1



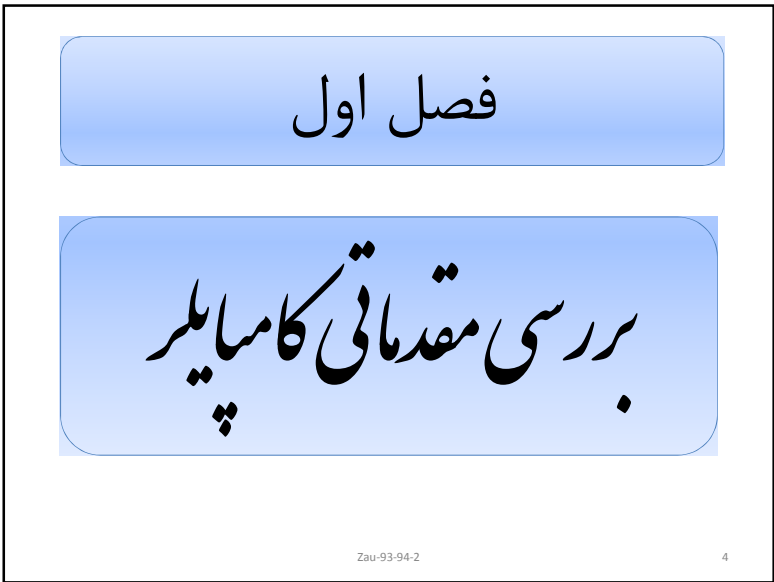
Zau-93-94-2

۲



Zau-93-94-2

۳



Zau-93-94-2

4

چرا درس اصول طراحی کامپایلر را می خوانیم؟

- ❖ اطلاعات عمومی از پشت صحنه برای یک مهندس نرم افزار خوب
- ❖ افزایش فهم از نحو زبانهای مختلف
- ❖ مشاهده نحوه نحوه تولید کد ماشین از یک زبان
- ❖ طراحی زبان خوب
- ❖ ابزارهای جدید نیازمند زبانهای مخصوصی می باشند
- ❖ مزایای کسب و کارهای جدید نیازمند زبانهای با دامنه مشخصی می باشند.



Zau-93-94-2

دسته بندی نرم افزارها



Zau-93-94-2

۶

ترجمه و مترجم



ترجمه:

- ❖ عبارتست از انتقال یک ساختار از یک زبان مبدأ به زبان دیگری به نام زبان مقصد.

مترجم:

- ❖ برنامه ای است که یک برنامه ی نوشته شده به زبان سطح بالا را در ورودی دریافت می دارد و برنامه ای به زبان دیگر به زبان مقصد را در خروجی تولید می کند.

Zau-93-94-2

۷

انواع مترجم

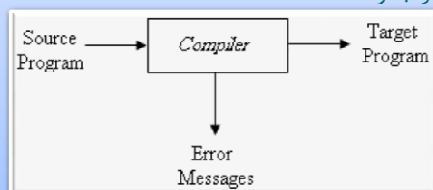
- ❖ کامپایلر (Compiler)
- ❖ مفسر (Interpreter)
- ❖ اسمبلر (Assembler)
- ❖ پیش پردازنده (Pre-Processor)

Zau-93-94-2

۸

کامپایلر چیست؟

- همگردان یا کامپایلر برنامه یا مجموعه‌ای از برنامه‌های کامپیوتری است که متنی از زبان برنامه نویسی سطح بالا (زبان مبدا) را به زبانی سطح پایین (زبان مقصد)، مثل اسمبلی یا زبان سطح ماشین، تبدیل می‌کند. خروجی این برنامه ممکن است برای پردازش شدن توسط برنامه دیگری مثل پیونددهنده مناسب باشد یا فایل متنی باشد که انسان نیز بتواند آنرا بخواند.

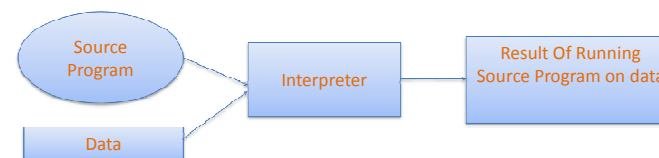


Zau-93-94-2

۹

مفسر چیست ؟

- ❖ برنامه ای است که برنامه منبع و داده های مورد نیاز آن را به عنوان ورودی دریافت کرده و در خروجی نتیجه اجرای برنامه منبع بر روی داده های ورودی را تولید می کند.



Zau-93-94-2

۱۰

تفاوت مفسر و کامپایلر چیست ؟

- ❖ مفسر برنامه ای است که همانند کامپایلر وظیفه ترجمه را به عهده دارد با این تفاوت که به جای اینکه برنامه ها را به صورت یکجا ترجمه و تفسیر نماید، هر دستورالعمل را به صورت انفرادی ترجمه و اجرا می کند. مفسرها اغلب کوچکتر از کامپایلرها هستند با این وجود سرعت اجرای یک برنامه تفسیر شده کندتر از یک برنامه کامپایل شده است.

- ❖ هرچند هر دو عمل ترجمه را انجام می دهند اما کامپایلر به صورت یکجا برنامه مبدأ را ترجمه می نماید و یک کد میانی بوجود می آید و سپس آن کد در پردازنده بار می شود و آن به صورت یکجا اجرا می شود. در مفسر چیزی به نام برنامه به زبان ماشین وجود ندارد.

Zau-93-94-2

۱۱

معایب مفسر

- ✓ سرعت اجرای برنامه پایین
- ✓ نیاز به مفسر
- ✓ نیازمند دسترسی به کد منبع

مزایای مفسر

- ✓ پیاده سازی آسان
- ✓ قابلیت حمل بالا
- ✓ سهولت اشکال زدایی
- ✓ قابلیت انعطاف بالا

Zau-93-94-2

۱۲

Compiler versus Interpreter

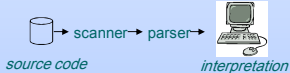
Compiler

translates to machine code



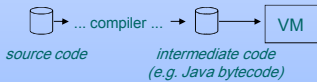
Interpreter

executes source code "directly"



- statements in a loop are scanned and parsed again and again
- source code is translated into the code of a virtual machine (VM)
- VM interprets the code simulating the physical machine

Variant: interpretation of intermediate code



Zau-93-94-2

۱۳

مفسر و کامپایلر

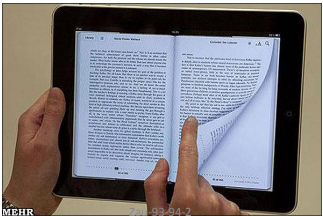
	مفسر	کامپایلر
حجم برنامه	کم	زیاد
سرعت اجرای برنامه	کم	زیاد
نحوه اجرا	خط به خط هم ترجمه هم اجرا	یک بار ترجمه یک بار اجرا
کدمیانی	نداریم	داریم
اجراهای بعدی	باید مجدداً ترجمه شود	نیاز به ترجمه مجدد نداریم
خطایابی	در حال نوشتن کد انجام می شود	پس از نوشتن کل کد برنامه

Zau-93-94-2

۱۴

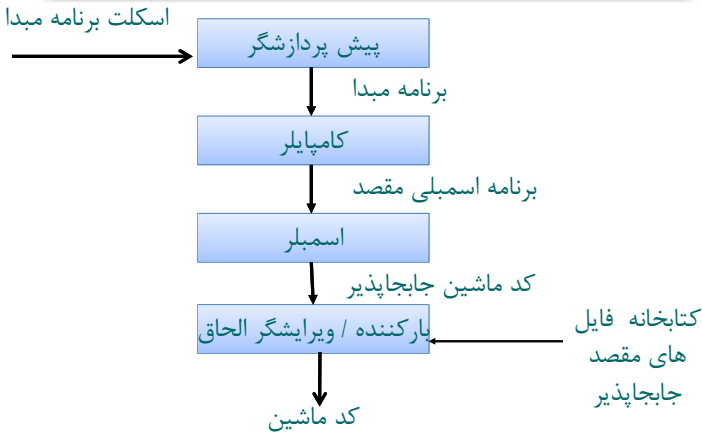
پیش پردازشگر

- ❖ جمع آوری ماژول های برنامه مبدا موجود در فایل های جداگانه
- ❖ تبدیل بخش های خلاصه شده بنام درشت دستورات به احکام زبان مبدا



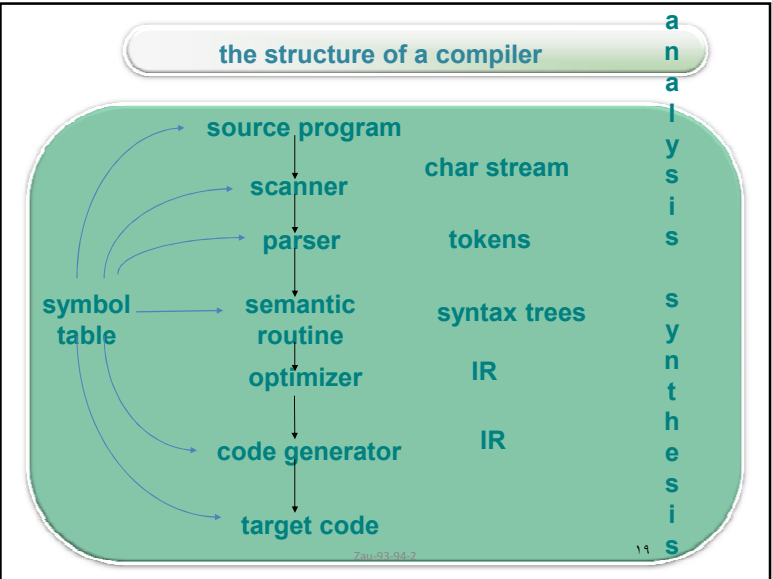
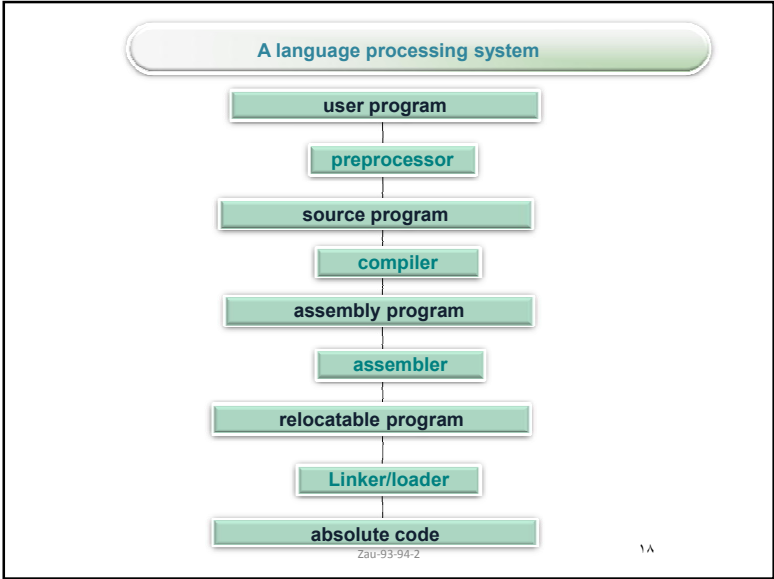
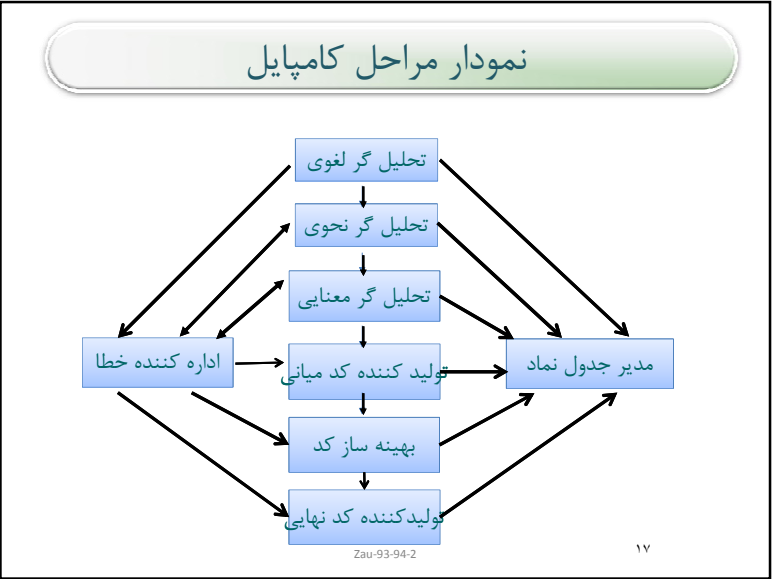
۱۵

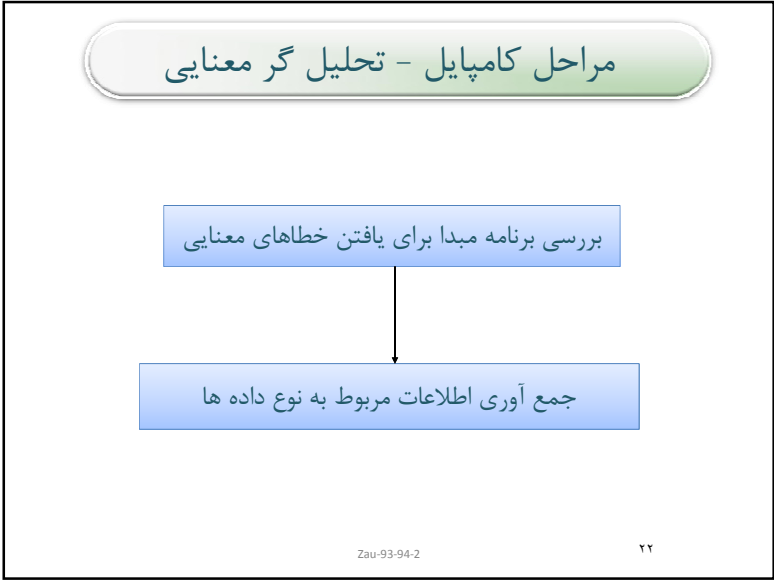
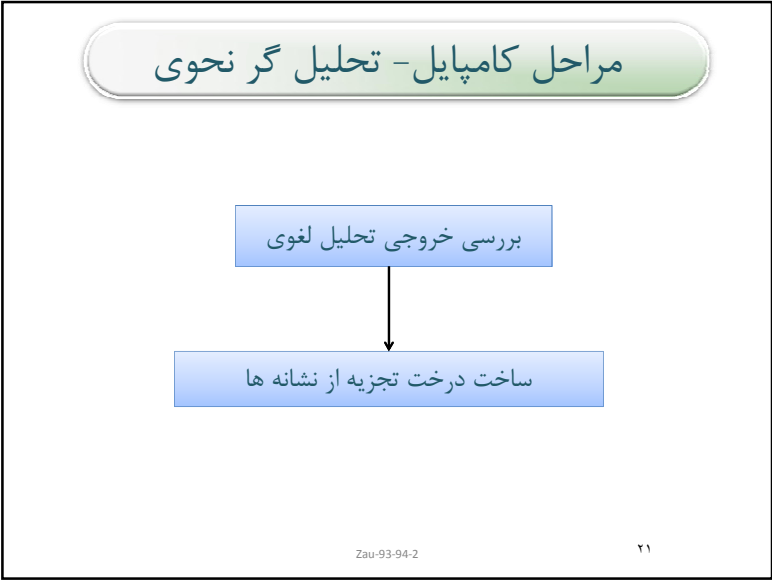
ارتباطات در سیستم پردازش زبان



Zau-93-94-2

۱۶

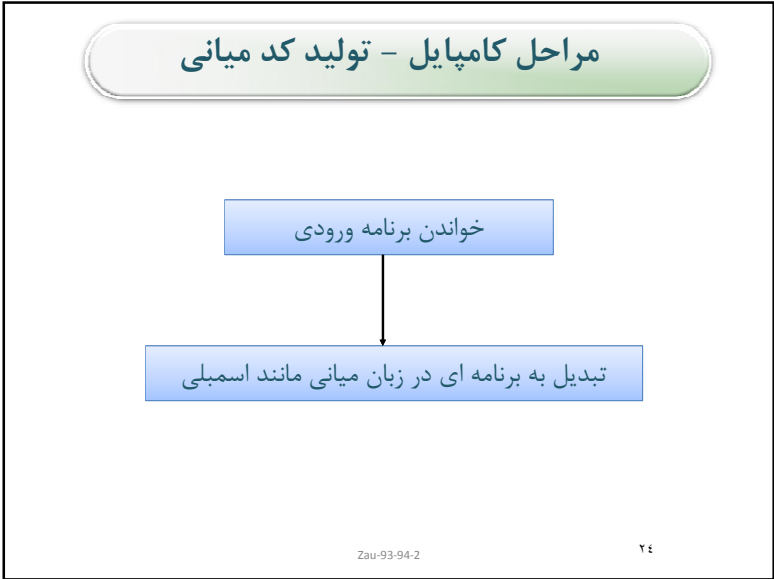




سه فاز تحلیل در عمل کامپایل

تشخیص نشانه ها	تحلیل خطی (تحلیل لغوی یا پویش)
گروه بندی نشانه های برنامه مبدا به جملات گرامری	تحلیل سلسله مراتبی (تحلیل نحوی یا تجزیه)
بررسی خطاهای معنایی برنامه	تحلیل معنایی

Zau-93-94-2 ۲۳



مراحل کامپایل - بهینه ساز کد

بهینه کردن کد میانی (حذف متغیرهای میانی غیر ضروری)

سرعت بخشیدن به تولید کد نهایی

Zau-93-94-2

۲۵

مراحل کامپایل - تولید کننده کد نهایی

تبدیل کد میانی بهینه به کد جابجاپذیر یا اسمبلی

تعیین مکانهای حافظه برای متغیرهای برنامه

انتساب متغیرها به ثبات های ماشین

Zau-93-94-2

۲۶

مراحل کامپایل - مدیریت جدول نماد

تعریف
ساختمان داده ای شامل
رکورد برای شناسه و
میدان هایی برای صفات آن

هدف

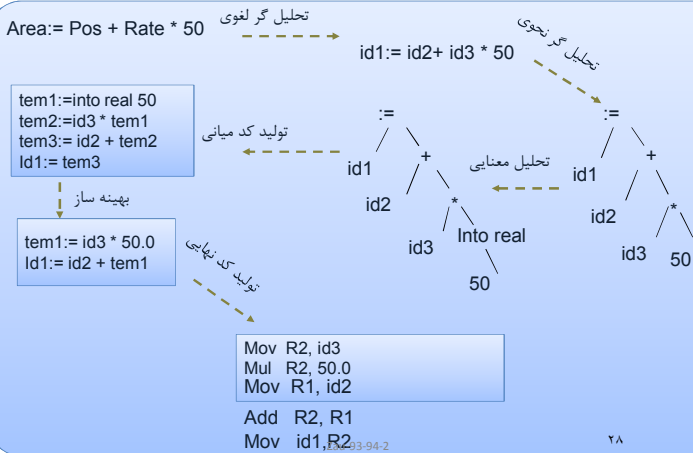
فراهم کردن شناسایی سریع رکورد شناسه
به منظور ذخیره و بازیابی داده هایش

Zau-93-94-2

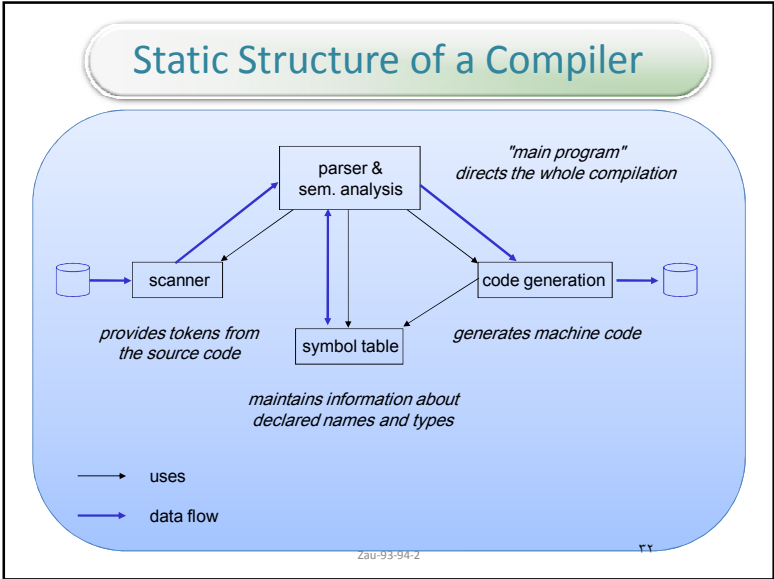
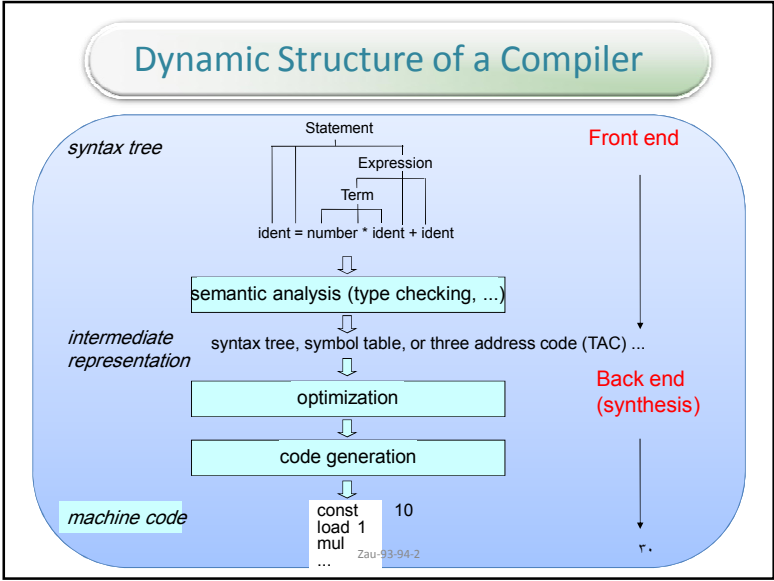
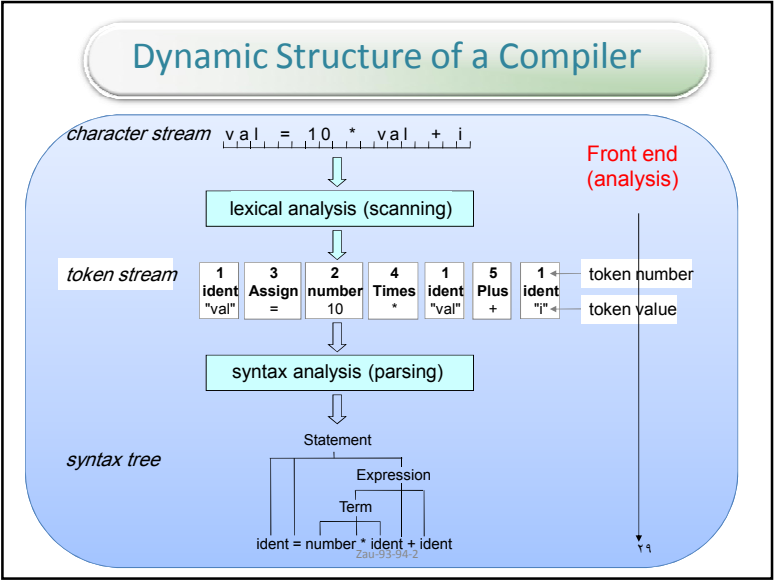
۲۷

مثال از مراحل کامپایل:

عبارت $Area := Pos + Rate * 50$



۲۸



وظایف تحلیلگر لغوی

- ۱- دریافت دنباله ای از کاراکترها
- ۲- تشخیص لغات
- ۳- تشخیص توکن ها و ارسال دنباله ای از توکن ها به تحلیلگر نحوی
- ۴- پردازش جاهای خالی (حذف جاهای خالی)
- ۵- پردازش توضیحات
- ۶- درج نام و نوع لغات در صورت امکان در جدول نمادها
- ۷- کشف ، گزارش و ترمیم خطاهای لغوی

Zau-93-94-2

۳۳

وظایف تحلیل گر نحوی

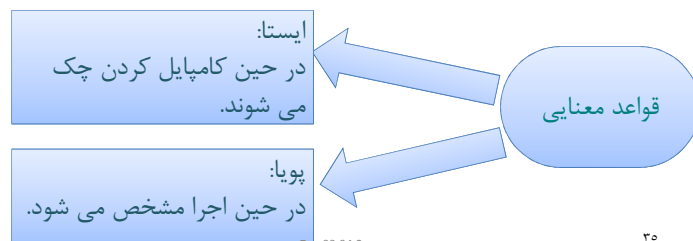
- ۱- دریافت دنباله ای از توکن ها از تحلیل گر لغوی
- ۲- بررسی برنامه ورودی از نظر دستوری و تطابق با syntax زبان برنامه سازی
- ۳- ساخت درخت تحلیل نحوی (pars tree)
- ۴- کشف ، گزارش و ترمیم خطاهای نحوی
- ۵- درج نوع بعضی لغات که درقسمت قبلی تشخیص داده نشده اند و یا غلط تشخیص داده شده اند
- ۶- ارسال درخت تحلیل به تحلیل گر معنایی

Zau-93-94-2

۳۴

قواعد معنایی

❖ اغلب زبان های برنامه نویسی قواعدی فراتر از لغوی و نحوی را برای خود تعریف می کنند که به آنها قواعد معنایی گویند.



Zau-93-94-2

۳۵

وظایف تحلیلگر معنایی

- ❖ ساختار کد میانی در این مرحله شکل می گیرد.
- ❖ پلی بین تجزیه تحلیل و ترکیب می باشد.
- ❖ عملکرد واقعی تحلیل گر معنایی با نگرش به نوع زبان و سازمان منطقی کامپایلر متفاوت می باشد .
- ❖ متداولترین وظایف تحلیلگر معنایی :
 - ❖ نگهداری جدول نمادها
 - ❖ درج اطلاعات ضمنی
 - ❖ کشف خطا
 - ❖ پردازش ماکرو و عملیات زمان ترجمه

Zau-93-94-2

۳۶

ماکرو

❖ ماکرو نام مخففی برای مجموعه ای از دستورالعمل ها، راهنماها یا ماکروهای دیگر است که یکبار نوشته می شود و به هر تعداد دفعات لازم قابل استفاده است. (مشابه زیربرنامه)

❖ اسمبلر هنگام ترجمه برنامه در مواجهه با نام ماکرو دستورات معادل را قرار می دهد.

Zau-93-94-2

۳۷

تولید کد میانی

❖ خیلی به زبان ماشین نزدیک است.

❖ این کد مستقل از زبان ماشین مقصد خواهد بود.

❖ این کد به دو روش زیر نگهداری می شود:

❖ شبه کد اسمبلی

❖ درختی

```
mov     ecx, ecx
cld
PrintLoop:
mov     dl, [esi]
inc     esi
inc     esi
mov     ah, 2
int     21h
```



Zau-93-94-2

۳۸

بهینه سازی کد میانی و تولید کد نهایی

❖ بهینه سازی: هدف از بهینه سازی تقلیل حجم کد و افزایش سرعت عمل کامپایلر می باشد. به همین منظور این واحد کد میانی تولید شده را مورد تحلیل قرار می دهد و سعی می شود با تکنیک های مختلف نظیر کار بر روی گرافها و غیره، حجم برنامه را کاهش دهد.

❖ کدنهایی: کار این واحد ترجمه نمایش میانی بهینه سازی شده ی برنامه ی مبدأ به کد ورودی ماشین مقصد می باشد.

Zau-93-94-2

۳۹

جدول نمادها

❖ ذخیره سازی نام ها و شناسه ها

❖ جمع آوری اطلاعات درباره مشخصات هر شناسه (نام؛ آدرس حافظه؛ محلی از برنامه که در آن تعریف شده؛ خط برنامه؛ نوع متغیر؛ در مورد تابع تعداد و نوع پارامترهای و روش ارسال پارامترها)

❖ تعامل بسیار زیاد با کامپایلر دارد بنابراین امکان دستیابی و به روز رسانی سریع بایستی فراهم باشد.

❖ در مرحله تحلیل لغوی کلیه شناسه ها موجود در جدول نمادها وارد می شوند. در مراحل دیگر؛ اطلاعات مربوط به شناسه ها به جدول اضافه خواهند شد و در مراحل کامپایل هر جایی که نیاز باشد از این اطلاعات استفاده خواهد شد.

Zau-93-94-2

۴۰

مدیریت خطاها

❖ انواع خطاها :

- خطای لغوی
تماما در زمان کامپایل توسط اسکنر شناسایی می شوند.
- خطای نحوی
تماما در زمان کامپایل توسط پارسر شناسایی می شوند.
- خطای معنایی
می توانند در زمان کامپایل یا زمان اجرا شناسایی شوند.
- خطای منطقی
در زمان اجرا شناسایی می شوند

Zau-93-94-2

۴۱

مدیریت خطاها

❖ کامپایلرهای قدیمی :

- تشخیص به موقع خطا و اطلاع به برنامه نویس (کامپایلرهای قدیمی)
- افزایش زمان توسعه نرم افزار

❖ کامپایلرهای جدید :

- سعی در رفع خطا
- تشخیص بیشتر خطا در یک بار کامپایل و گزارش به برنامه نویس
- ❖ قدرت خطاپرداز به روشهای رفع خطای به کار گرفته شده وابسته است .
- ❖ معمولا اکثر خطاها از نوع نحوی هستند.

Zau-93-94-2

۴۲

جلوبندی کامپایلر و عقب بندی کامپایلر (Front end & Back end)

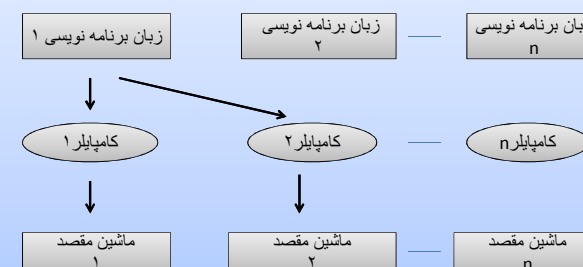
میخواهیم یک برنامه به زبان C را بر روی کامپیوترهای مختلف همانند (SUN, IBM, VAX و...) اجرا کنیم.
در این صورت برای هر نوع کامپیوتر از انواع بالا (ماشین مقصد) یک نوع کامپایلر جداگانه باید بسازیم.
در حالت کلی اگر n زبان برنامه سازی سطح بالا (C, Pascal, Fortran و...) داشته باشیم و k تعداد معماریهای مختلف ماشین مقصد باشد (SUN, IBM و...) در این صورت به $(N * K)$ کامپایلر نیاز داریم.

Zau-93-94-2

۴۳

جلوبندی کامپایلر و عقب بندی کامپایلر (ادامه)

برای حل این مشکل از تقسیم کامپایلر به عقب بندی و جلو بندی استفاده میکنیم:



M.Bazargani ۹۲-۹۴-۲

Zau-93-94-2

۴۴

جلوبندی کامپایلر و عقب بندی کامپایلر (ادامه)

برای فهم جلوبندی و عقب بندی کامپایلر به یک مثال توجه کنید فرض کنید یک متن خبری به زبان روسی قرار است به تمام زبانهای زنده دنیا که تعداد آن n است ترجمه شود، چند مترجم نیاز داریم؟

❖ به $n-1$ زبان مترجم نیاز داریم.

حال فرض کنید قرار است یک متنی را از هر زبانی به هر زبان دیگری ترجمه کنیم. در این صورت تعداد مترجم های مورد نیاز چقدر است؟

❖ $n(n-1)/2$ مترجم مورد نیاز است.

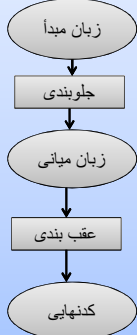
تعریف جلوبندی کامپایلر و عقب بندی کامپایلر

❖ بخشی از کامپایلر که وظیفه ترجمه برنامه مبدأ به زبان میانی را برعهده دارد جلوبندی کامپایلر نام دارد.

❖ بخشی از کامپایلر که وظیفه ی ترجمه ی زبان میانی به کد نهایی را برعهده دارد عقب بندی کامپایلر نام میگیرد.

جلوبندی کامپایلر و عقب بندی کامپایلر (ادامه)

با استفاده از این روش اگر n زبان مبدأ و k ماشین مقصد داشته باشیم به چند جلوبندی و چند عقب بندی نیاز داریم؟



❖ به n جلوبندی و k تا عقب بندی نیاز داریم.

❖ به $n+k$ تا برنامه ی میانی نیاز داریم.

جلوبندی کامپایلر و عقب بندی کامپایلر (ادامه)

جلوبندی کامپایلر وابسته به زبان مبدأ و عقب بندی آن وابسته به زبان مقصد خواهد بود.

