کامپایلر
مقدمات

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

اعلانات

مدرس: محسن هوشمند

بخش تمرین و کمک: محمدمهدی علی‌جانی

نشانی دسترسی به مانهٔ درس:

https://iasbs.ac.ir/~mohsen.hooshmand/courses/9900/1/compiler991.html

Compile: to compose out of materials from other documents. *Merriam-Webster*

معادل‌های COMPILER

فرهنگستان

مترجم	[رایانه و فنّاوری اطلاعات]	برنامه‌ای که دستورهای نوشته‌شده به زبانی سطح بالاتر را رمزگشایی و برنامه‌ای به زبان هم‌گذاری تولید کند
گردآورنده	[علوم کتابداری و اطلاع‌رسانی]	شخصی که آثار نوشتاری مختلف چاپی یا دستنویس را برای چاپ در مجموعه‌ای منظم گرد بیاورد

واژه‌نامه کامپیوتر
کامپایلر، همگردان، مترجم

مراجع

[ببرسبز] A. Appel, M. Ginsburg, "**Modern Compiler Implementation in C,**" Cambridge University Press, 1998

[ببرسرخ] ----------, "**Modern Compiler Implementation in Java,**" Cambridge University Press, 2002

[اپل‌م‌ل] ----------, "**Modern Compiler Implementation in ML,**" Cambridge University Press, 1998

[اژدرها] A. V. Aho, et al., "**Compilers: Principles, techniques, & Tools,**" Pearson, 2nd ed, 2007

[اسکات] M. Scott, "**Programming Language Pragmatics,**" Morgan Kaufmann, Elsevier, 4th ed, 2016.

[فیشر] C. N. Fischer, et al., "**Crafting a Compiler,**" Pearson, 2010.

[کوپر] K. D. Cooper, L. Torczon "**Engineering a compiler,**" Morgan Kaufmann, Elsevier, 2nd ed, 2012.

[کوپر۷] K. D. Cooper, L. Torczon "**Engineering a compiler,**" Morgan Kaufmann, Elsevier, 2007.

[جادوگر] H. Abelson, J. Sussman, J. Sussman, "**Structure and Interpretation of Computer Programs,**" MIT, 2nd ed, 1996.

جزوهٔ غلامرضا ثانی، هیئت علمی دانشکده مهندسی کامپیوتر دانشگاه شریف

کمکی

[ریچی] B. W. Kernighan • D. M. Ritchie, "**The C Programming Language,**" Prentice Hall, 2nd ed, 1988.

[فرهنگستان]

[انوری] ح. انوری و دیگران، «**فرهنگ بزرگ سخن،**» انتشارات سخن، ۱۳۹۲

[رانکوهی] م.ت. روحانی‌رانکوهی، «**فرهنگ داده،**» انتشارات جلوه،

Digital filters [7]–[10]: sophisticated filter banks [7] were designed to recognize QRS complexes in which they analyzed the positions and magnitudes of sharp waves and used a special digital band-pass filter to reduce the false detection of ECG signals in the MIT-BIH database [11]. The difference operation method (DOM) [8] scheme including two stages was proposed: the first stage was to find the point R by applying the difference equation operation to an ECG signal, then the second stage looked for the points Q and S based on the point R to find the QRS complex. The work [9] used some special digital filters to detect and classify ECG signal in time or frequency domain. Slope- and peak-sensitive band-pass filters were employed for the detection [10]. The morphological smoothing further improved its performance.

Wavelet transform (WT) [12]–[16]: the transform yields a time-scale representation similar to the time-frequency representation of the short-time Fourier transform (STFT) [12], while the WT uses a set of analyzing functions that allows a variable time and frequency resolution for different frequency bands [13]. By the multiscale feature of WT, the QRS complex can be distinguished from high P or T waves, noise, and baseline drift. The dyadic discrete WT (DWT) was usually implemented using a dyadic filter bank where the filter coefficients were directly derived from the wavelet function [14]. The WT based on the adaptive threshold [15] and WT based on multi-lead ECG [16] were evaluated on the QT database [17].

Adaptive matched filters [18]–[22]: a two-stage successive cancellation algorithm that sequentially separates

Bayesian Real-Time QRS Complex Detector for Healthcare System

ارزیابی

تمرین‌ها
- کتبی
- عملی

مقاله

امتحان

- خلق دانش
- کپی‌پیست (گرفتن‌گذاشتن) نمره‌ای تعلق نمی‌گیرد

اعلام منابع هر تمرین

اعلانات

تقلب: قانون دانشگاه

دیرکرد مقبول: پنج روز که می‌توان به یک یا چند مهلت تحویل تقسیم کرد.

- با اعلام

امتحان دوباره به هر دلیلی گرفته نخواهد شد

اعلانات

نحوه ارسال تمرین‌ها، پروژه‌ها، دیگر موارد

- ا-نامه compiler.iasbs.13991@gmail.com
  - عنوان: «کامپایلر- تمرین سری اول»
  - فایل متنی: قالب پی‌دی‌اف
  - نام فایل: C-T#-Afshar-Mahmoud_Sotode-Morteza.pdf
  - نام فایل: C-P#-Afshar-Mahmoud_Sotode-Morteza.pdf

  - دیگر فایل‌ها
  - نام فایل: C-T#-Afshar-Mahmoud_Sotode-Morteza.zip

AUTOMATA STUDIES و دیگر مقالات

“Representation of Events in Nerve Nets and Finite Automata,” *By S. C . Kleene*

“Probabilistic Logics and the Synthesis of Reliable Organisms From Unreliable Components,” *By J. von Neumann*

“Some Universal Elements for Finite Automata,” *By M. L. Minsky*

“Gedanken-Experiments on Sequential Machines,” *By Edward F. Moore*

“A Universal Turing Machine with Two Internal States,” *By Claude E. Shannon*

“A Note on Universal Turing Machines,” *By M. D. Davis*

“The Inversion of Functions Defined by Turing Machines,” *By John McCarthy*

“Computability by Probabilistic Machines,” *By K. de Leeuw, E. F. Moore, C. E. Shannon, and N. Shapiro*

“The Epistemological Problem for Automata,” *By D. M. MacKay*

“The Design of Conditional Probability Computers,” *By Albert M. Uttley*

“On the Translation of Languages from Left to Right,” By Donald E. Knuth

“Semantics of Context-Free Languages,” by Donald E. Knuth

“The Cost of Lexical Analysis,” by W. M. Waite

“Multibox parsers,” by Lev. J. Dyadkin

“Multibox Parsers: No More Handwritten Lexical Analyzers,” by Lev. J. Dyadkin

AUTOMATA STUDIES و دیگر مقالات – ادامه

“The Complexity of Songs,” by D. E. Knuth, 1984

“Semantics of Context-free Languages,” by D. Knuth, 1974

اختصارات

اژدها: کتاب اهو و همکاران

ت‌ب‌پ: تجزیه بالا به پائین

ت‌پ‌ب: تجزیه پائین به بالا

جادوگر: کتاب ابلسن و همکاران

خم: خودکاره متناهی

خمم: خودکاره متناهی معین

خمن:خودکاره متناهی نامعین

ع‌ب: عبارت منظم

فیشر: کتاب فیشر و همکاران

ص‌ب‌ن: صورت بکوس-نائور

کوپر: کتاب کوپر و همکاران

مام: مستقل‌از متن

م‌ک‌م: مولد کد میانی

ممج: ماشین مجازی جاوا

Σ: مجموعه متناهی حروف الفباء دستور

BNF: صورت بکوس-نائور

R: مجموعه قوانین تولید دستور

S: متغیر آغاز دستور

V: مجموعه متناهی متغیرهای دستور

مقدمه

نقش کامپیوتر در زندگی روزانه

خیزش اینترنت

- کامپیوتر و نرم‌افزارهای نصب شده بر آن موجب ارتباط
- کامپیوترهای نشانده embedded
  - دریافت ایمیل در جیب شما
- موجب فعالیت‌های جدید
  - از بازی‌ها تا شبکه‌های اجتماعی
- ابرکامپیوترها
  - پیش‌بینی هوا

مقدمه

تمامی این نوآوری‌ها مبتنی بر برنامه‌های نرم‌افزار

- تولید ابزارهای مجازی بر روی انتزاعی فراهم‌شده توسط سخت‌افزار

ترجمه تقریبا تمامی نرم‌افزارها با ابزاری به نام «**کامپایلر**»

کامپایلر

- برنامه کامپیوتری ترجمه‌گر برنامه‌های دیگر که آنها را برای اجرا آماده می‌سازد

هدف این درس:

عرضهٔ فنون بنیادی سازندهٔ کامپایلر

چالش‌های پیش‌روی ساخت کامپایلر و الگوریتم‌های استفاده شده

انواع زبان

زبان ماشین

- به صورت اعداد دودوئی
- متفاوت از ماشین به ماشین

زبان اسمبلی

- استفاده از کلمات اختصاری

زبان سطح بالا

- نزدیک‌تر به زبان انسان
- دارای ساختار و امکانات بیشتر

تاریخ کامپایل

اولین استفاده از کلمه کامپایلر
- گریس مورای هوپر

به مثابهٔ برنامه‌نویسی خودکار
- تردید در موفقیت

هنوز هم شناخته شدن مترجم‌های زبان برنامه‌نویسی به کامپایلر

اولین‌ها
- فرترن

ابتدا ساختارهای اقتضائی و ضمنی
- امروزه روش‌مند

اولین زبان کامپایل‌شده روی بسترهای مختلف
- کوبول

ادامهٔ مطالعات پویش‌گر و تجزیه در دهه ۴۰ و ۵۰ شمسی

تاریخ کامپایل

زبان‌های سطح بالا

حروف‌چین
- تک و لتک

پست‌اسکریپت

متمتیکا

مرور – برنامه کامپیوتری

برنامه کامپیوتری

- دنباله‌ای از عملیات‌های انتزاعی
- نوشته شده با زبانی برنامه‌نویسی
  - زبانی صوری که جهت اظهار محاسبات طراحی شده است
- دارای معنا و ظاهر مشخص
  - بر عکس زبان طبیعی
- جهت سلاست و ایجاز و وضوح
  - Expressiveness and conciseness and clarity
- زبان طبیعی اجازه به ابهام
  - زبان برنامه‌نویسی سعی بر دوری از ابهام
  - برنامه مبهم، معنا ندارد

مرور - *ادامه*

زبان برنامه‌نویسی

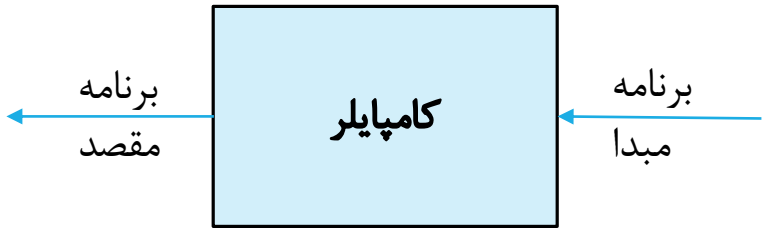
- دنباله‌ای از عملیات‌های انتزاعی
- جهت اظهار دنباله‌ای از عملیات‌ها

پردازنده کامپیوتری

- اجرای دنباله‌ای از عملیات‌ها
- عملیات‌های پردازنده دارای انتزاع کمتر نسبت به انتزاع زبان‌های برنامه‌نویسی
- مثال
  - دستور زبان سطح بالای چاپ: مختصر و واضح
  - دستور معادل ماشین: صدها عملیات ماشین تا اجرا ممکن شود

ابزاری که چنین ترجمه‌ای را ممکن می‌کند: کامپایلر

مرور - *ادامه*



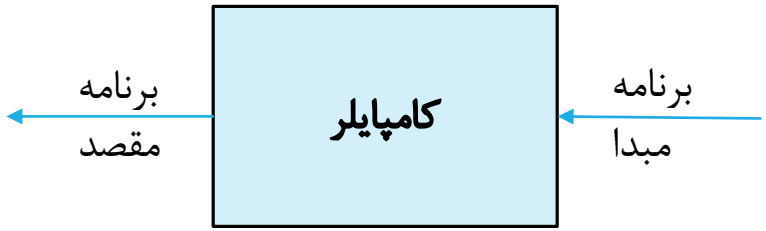
زبان مبدا

- سی
- سی++
- فرترن
- جاوا

زبان مقصد

- مجموعه دستورالعمل **instruction set** پردازنده‌ای
  - مجموع دستوراتی که پردازنده‌ای در اختیار می‌گذارد
  - طراحی مجموعه دستورات را معماری مجموعه دستورالعمل ISA می‌گویند.
- گاهی اوقات زبان به زبان برنامه‌نویسی انسان-محور ترجمه می‌شود
  - مثال تبدیل زبان به زبان سی،
    - چون کامپایلر سی را اکثر کامپیوترها در اختیار دارند
    - ترجمهٔ مبدا به مبدا **source to source translators**

مرور - ادامه



مفسر interpreter

- دریافت ورودی به عنوان کد اجرائی
- تولید خروجی از هر خط کد
- مثال Perl و Scheme و APL

بعضی زبان‌ها هر دو

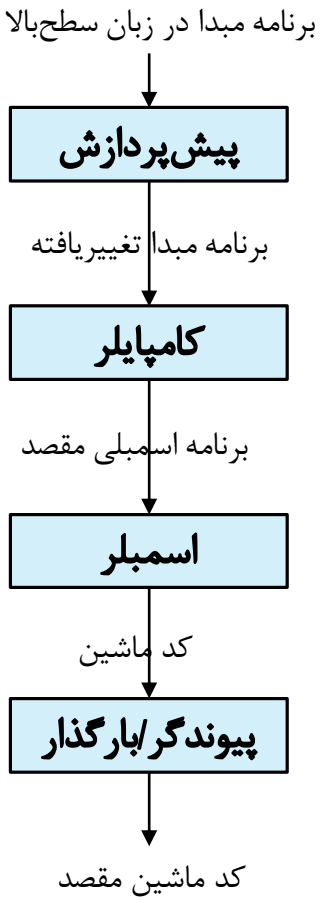
- جاوا
  - ترجمه زبان به کدبایت bytecode (نمایشی موجز) با هدف کاهش تعداد پیاده کردن کاربردهای جاوائی
  - اجرای کدبایت در ماشین مجازی جاوا JVM مmج
  - مmج مفسر کدبایت

مرور - ادامه

ایجاد زبان مقصد اجراپذیر نیاز به اجزائی غیر از کامپایلر

- کد
  - تقسیم به ماژول‌های ذخیره شده در جاهای متفاوت
- پیش‌پردازش
  - افزودن ماکروها و فایل‌های افزوده شده
  - تحویل برنامه سرهم‌شده به کامپایلر
- تولید زبان اسمبلی در کامپایلر
  - پردازش زبان اسمبلی با اسمبلر
    - کد جابجاپذیر
    - کد مطلق
- برنامه‌بزرگ در قطعات ظاهر می‌شوند
  - پیوند کدها و کتابخانه‌ها و کد شی‌های مختلف با یکدیگر
  - فایل‌های کامپایل‌شده با اسمبل‌گرهای متفاوت
  - پیوندگر linker
- قرار دادن تمامی قطعات اجرا پذیر در حافظه
  - بارگذار loader
    - بخش از سیستم‌عامل
    - محاسبهٔ اندازه برنامه جهت ایجاد فضای حافظهٔ اضافی

gcc



سطح بالا یا پائین!

زبان سطح بالا

- ساده‌تر جهت برنامه‌نویسی
- ناکارآمدتر از زبان مقصد
- کندتر از زبان مقصد

زبان سطح پائین

- کنترل بیشتر بر رایانش
- تولید کد کارآمدتر
- سخت‌تر جهت برنامه‌نویسی
- حمل‌ناپذیرتر
- در معرض خطا
- نگهداری سخت‌تر

فواید مطالعه کامپایلر

برنامه پیچیده و بزرگ

تمرینی اساسی در مهندسی نرم‌افزار

- چگونگی ساخت مدل ریاضی درست و الگوریتم درست با حفظ تعادل بین تعمیم‌پذیری و قدرت با سادگی و کارائی

خردجهان علم رایانه

- استفاده عملی از
  - خودکاره متناهی و پشته‌ای: پویش و تجزیه scanning and parsing
  - الگوریتم‌های حرصی (تخصیص ثبات register allocation(
  - جستجوی یافتاری (هیورستیک) زمان‌بندی فهرست List scheduling
  - الگوریتم‌های گراف: حذف کد-مرده
  - برنامه‌نویسی پویا: انتخاب دستور
  - الگوریتم‌های نقطه ثابت: آناکاوی جریان داده
- در تعامل با
  - تخصیصی پویا، همزمان‌سازی، نام‌دهی، محلی Locality، مدیریت مراتب حافظه، زمان‌بندی لوله
  - کمتر نرم‌افزاری با چنین خصوصیاتی

فواید مطالعه کامپایلر- *ادامه*

- نقش بنیادی در علم رایانه
  - آماده‌سازی مسائل جهت حل با رایانه
  - درستی فرایند و کارایی کد حاصل تاثیر مستقیم بر ایجاد سیستم‌های بزرگ
  - خواندن موجب تایید نیست باید پیاده شود
- جزئی اساسی در اموزش علم رایانه

نمایشگر کاربرد موفق نظر در حل مسائل عملی

- استفاده از نظریه زبان صوری در
  - پویشگر و تجزیه‌گر (عبارات منظم و دستورات مستقل از متن)
  - جستجوی متن
  - فیلتر وب‌سایت
  - پردازش لغت
  - مفسر زبان-فرمان command-language
- نظریه مشبک (ترتیب) lattice theory و نظریه اعداد در
  - املا
  - تحلیل ایستا static analysis
- درخت‌ها
  - ساختار برنامه و ترجمه آن به کد شی

فواید مطالعه کامپایلر- *ادامه*

بعضی مسائل باز و حل‌ناشده

- طراحی نمایش میانی همه‌گیر، سطح‌بالا ریشه در پیچیدگی
- دستورات زمان‌بندی مبتنی بر حرصی
  - نیاز به تعاون و همکاری

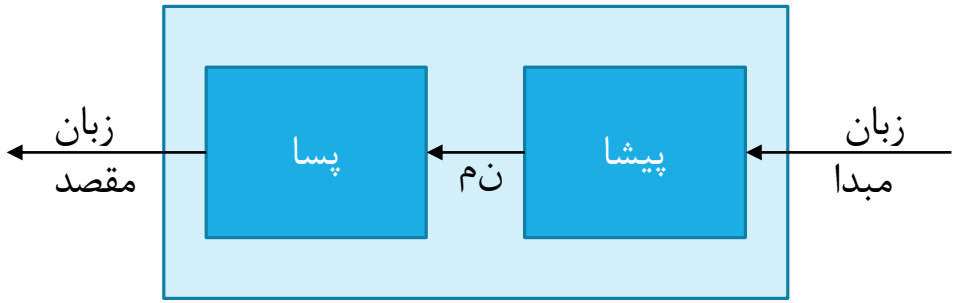
طراحی کامپایلر نیاز به تخصص در الگوریتم و مهندسی و برنامه‌ریزی planning

اصول بنیادی کامپایل‌گری

درستی correctness

- کامپایلر باید معنای برنامه کامپایل شده را حفظ کند
- کامپایلر باید برنامه ورودی را با روشی واضح بهبود دهد.

ساختمان کامپایلر

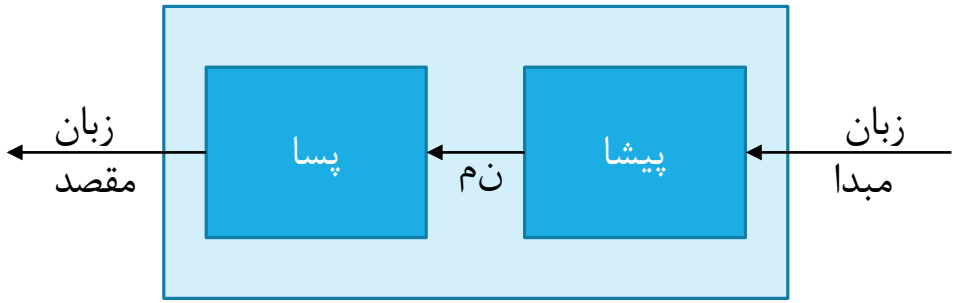


از سال ۱۹۵۵

فهم زبان مبدا و تبدیل کارکردهایش به زبان مقصد

- تقسیم به دو بخش پیشا و پسا (آنالیز و سنتز)
- پیشا
  - تمرکز بر فهم زبان مبدا
  - اطمینان از خوش‌فرمی کد
  - نگاشت کد به نمایش میانی
- پسا
  - تمرکز بر نگاشت برنامه به ماشین مقصد
  - نگاشت برنامه میانی به مجموعه دستورات و منابع متناهی ماشین مقصد
  - فرض بر عاری بودن نمایش میانی از خطای نحوی و معنائی
- نمایش میانی intermediate representation
  - مجهز بودن هر مرحله به نمایشی مشخص
  - نمایش‌های میانی متفاوت
  - بین هر دو فاز یک نمایش میانی

ساختمان کامپایلر

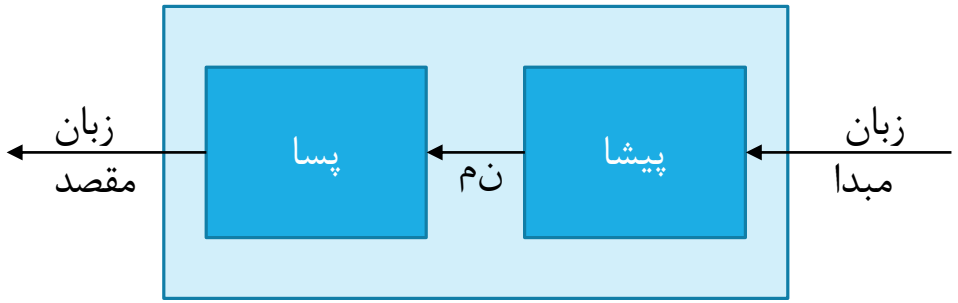


پیشا

تحلیل یا آنالیز

- تقسیم کد به قطعات سازنده و
- اعمال ساختار دستوری بر آن‌ها
- استفاده از این ساختار جهت ایجاد نمایش میانی
- در صورت تشخیص خوش‌فرم نبودن نحوی یا درست‌نبودن معنایی
  - ایجاد پیام واضح به کاربر جهت تصحیح کد
- جمع‌آوری اطلاع دربارهٔ برنامهٔ مبدأ و ذخیره در «جدول علامت»
- ارسال نمایش میانی و جدول علامت به پسا

ساختمان کامپایلر



پسا

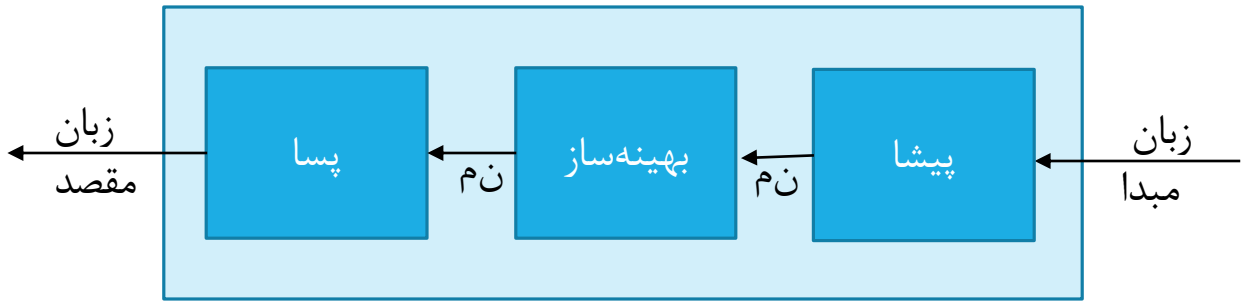
سنتز
- تولید برنامه مقصد مطلوب از نمایش میانی و اطلاعات جدول علامت

مزایای استفاده از پیشا و پسا
- سادگی طراحی
- استقلال پیشا از کد مقصد
- استقلال پسا از کد مبدا
- کاهش پیچیدگی
- بازاستفاده‌پذیری
- افزایش سرعت تولید کامپایل‌گر بابت سخت‌افزار جدید و زبان‌های جدید
  - n زبان برنامه‌نویسی و k ماشین متفاوت
  - nk کامپایلر جهت اجراء هر زبان بر هر ماشین

ساختمان کامپایلر/ادامه

نمایش میانی امکان افزودن فازهای بیشتر کامپایل

- فاز سوم بین پیشا و پسا
- بهینه‌ساز optimizer
  - جهت بهبود ترجمه
  - تحلیل و بازنویسی صورت میانی
  - ورودی آن برنامه نمایش میانی و تولید برنامه نمایش میانی متناظر از لحاظ معنائی
  - جهت بهبود آن
  - اهداف ممکن: انرژی کمتر



دسته‌بندی کامپایل‌گرها

بر اساس ترجمهٔ همه کامپایل‌گرها یکسان

- تفاوت صرفا در زبان مبدا و مقصد
- ولی بر اساس گزینه‌های توصیف خروجی متفاوت و امکان دسته‌بندی

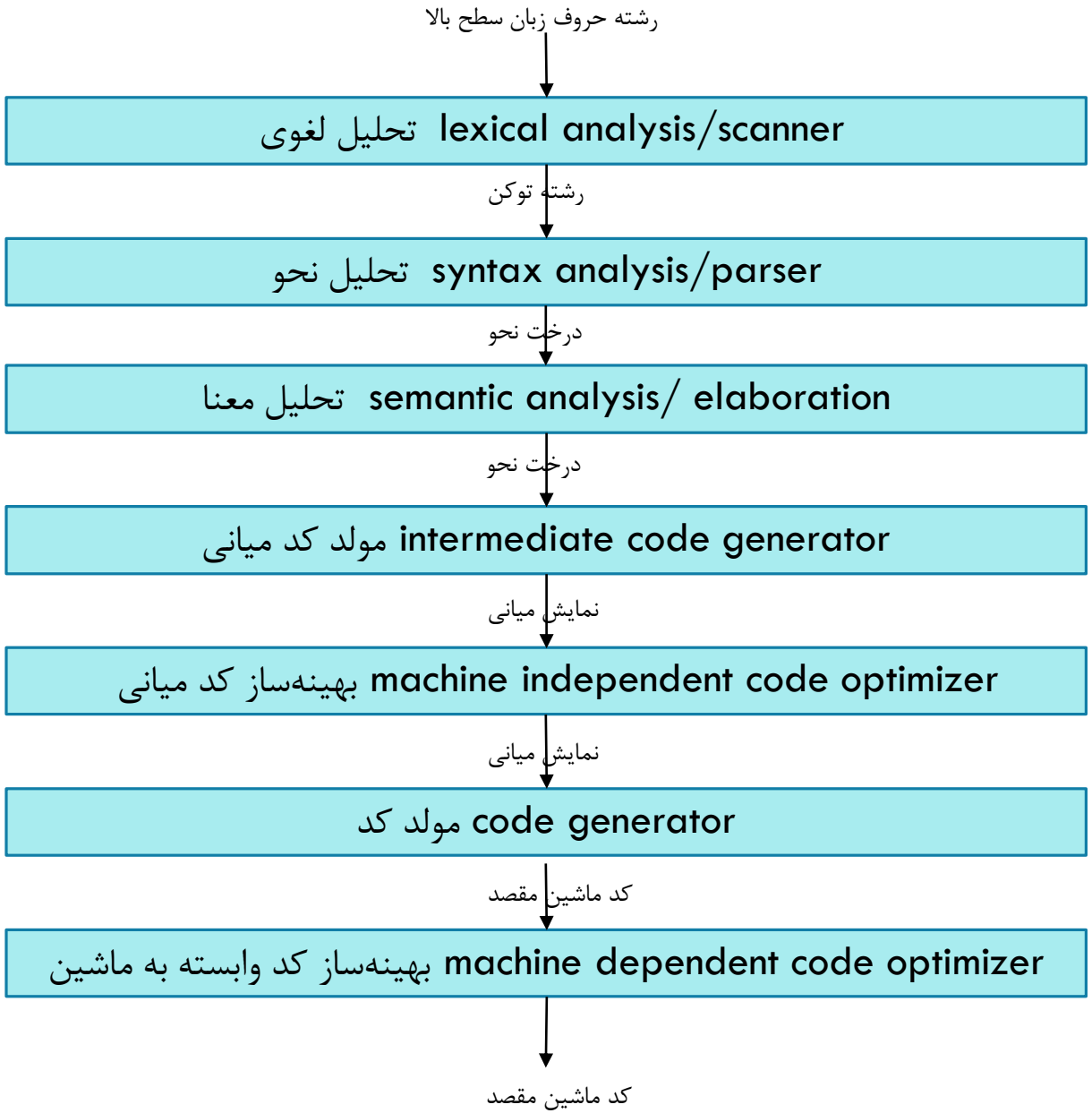
مبتنی بر نوع کد ماشین تولیدی

- کد ماشین خالص
  - تولید کد جهت ماشینی با مجموعه دستورات خاص
  - بدون توجه به سیستم عامل و یا کتابخانه‌ها
- کد ماشین افزوده
  - تولید کد با افزودن رویه‌های سیستم عاملی
  - فرترن
- کد ماشین مجازی
  - دستورهای مجازی
  - حمل‌پذیری
  - جاوا
  - اجرا با روش دقیقا سروقت JIT

مبتنی بر قالب کد مقصد تولیدی

- اسمبلی
  - ساده‌کردن و بخش‌بندی بیشتر ترجمه
  - نشانی‌دهی و دستورنویسی بر عهدهٔ اسمبل‌گر
  - ترجمهٔ خارجی
  - تولید زبان دیگر (سی)
- دودوئی جابجاپذیر relocatable
  - اجازهٔ کنترل بیشتر
  - ارجاع بیرونی و نشانی‌دهی مقیدنشده
  - نسبی
- دودوئی مطلق
  - آمادهٔ اجراء پس از اتمام کامپایل
  - نیاز به بازکامپایل در هر اجرای دیگر

ساختمان کامپایلر-ادامه



جدول علامت
Symbol table

خطاپرداز
Error handler

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	

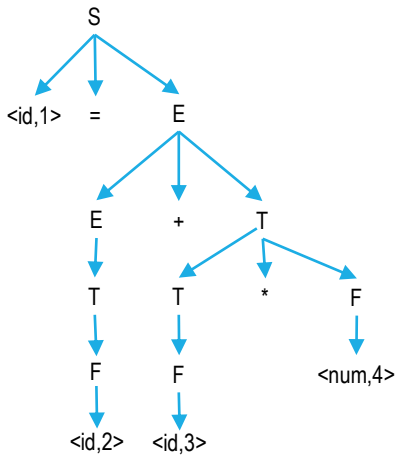
تحلیل نحوی

$S \rightarrow id = E$

$E \rightarrow E + T|E$

$T \rightarrow T * F|F$

$F \rightarrow num$



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

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