

به نام خدا

Compiler Design

مهدی حقوردی



دانشگاه اصفهان

مقدمه

معرفی

پردازشگرهای زبان

ساختار یک کامپایلر

مقدمه

- در این ارائه به بررسی طراحی کامپایلر می پردازیم.
- در این ارائه سعی شده است که با نشان دادن مثال و کدهای واقعی فهم قسمت های مختلف یک کامپایلر (یا مفسر) برای شما ساده شود.
- در ابتدا به معرفی و بررسی قسمت های مختلف یک کامپایلر پرداخته می شود.
- در حین معرفی و توضیح، مثال هایی از آن قسمت نشان داده می شود،
- و در پایان دو کتاب برای مطالعه ی عمیق روی کامپایلرها و مفسرها معرفی می شوند.

- در این ارائه به بررسی طراحی کامپایلر می پردازیم.
- در این ارائه سعی شده است که با نشان دادن مثال و کدهای واقعی فهم قسمت های مختلف یک کامپایلر (یا مفسر) برای شما ساده شود.
- در ابتدا به معرفی و بررسی قسمت های مختلف یک کامپایلر پرداخته می شود.
- در حین معرفی و توضیح، مثال هایی از آن قسمت نشان داده می شود،
- و در پایان دو کتاب برای مطالعه ی عمیق روی کامپایلرها و مفسرها معرفی می شوند.

- در این ارائه به بررسی طراحی کامپایلر می پردازیم.
- در این ارائه سعی شده است که با نشان دادن مثال و کدهای واقعی فهم قسمت های مختلف یک کامپایلر (یا مفسر) برای شما ساده شود.
- در ابتدا به معرفی و بررسی قسمت های مختلف یک کامپایلر پرداخته می شود.
- در حین معرفی و توضیح، مثال هایی از آن قسمت نشان داده می شود،
- و در پایان دو کتاب برای مطالعه ی عمیق روی کامپایلرها و مفسرها معرفی می شوند.

- در این ارائه به بررسی طراحی کامپایلر می پردازیم.
- در این ارائه سعی شده است که با نشان دادن مثال و کدهای واقعی فهم قسمت های مختلف یک کامپایلر (یا مفسر) برای شما ساده شود.
- در ابتدا به معرفی و بررسی قسمت های مختلف یک کامپایلر پرداخته می شود.
- در حین معرفی و توضیح، مثال هایی از آن قسمت نشان داده می شود،
- و در پایان دو کتاب برای مطالعه ی عمیق روی کامپایلرها و مفسرها معرفی می شوند.

- در این ارائه به بررسی طراحی کامپایلر می پردازیم.
- در این ارائه سعی شده است که با نشان دادن مثال و کدهای واقعی فهم قسمت های مختلف یک کامپایلر (یا مفسر) برای شما ساده شود.
- در ابتدا به معرفی و بررسی قسمت های مختلف یک کامپایلر پرداخته می شود.
- در حین معرفی و توضیح، مثال هایی از آن قسمت نشان داده می شود،
- و در پایان دو کتاب برای مطالعه ی عمیق روی کامپایلرها و مفسرها معرفی می شوند.

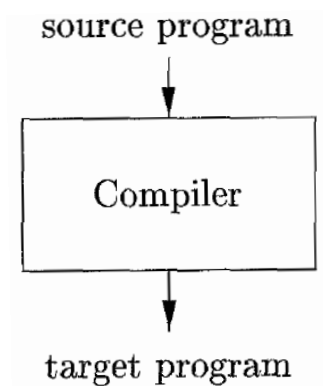
معرفی

- زبان‌های برنامه‌نویسی، نمادها و ابزاری برای توصیف محاسبات برای انسان‌ها و کامپیوترها هستند.
- جهانی که ما می‌شناسیم به زبان‌های برنامه‌نویسی وابستگی بسیار زیادی دارد، چون تمام نرم‌افزارهای روی کامپیوترها دنیا با زبانی نوشته شده‌اند.
- اما قبل از اینکه بتوانیم آنها را اجرا (run) کنیم، باید بتوانیم آنها را به حالتی تبدیل کنیم که پردازنده‌های ما بتوانند آنها را اجرا کنند.

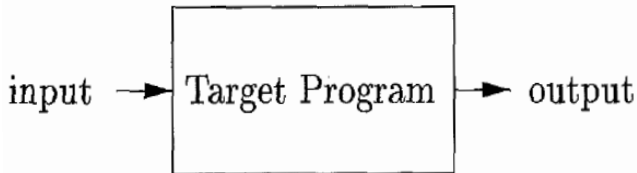
نرم‌افزاری که این کار را برای ما انجام می‌دهد، کامپایلر نام دارد.

پردازشگرهای زبان

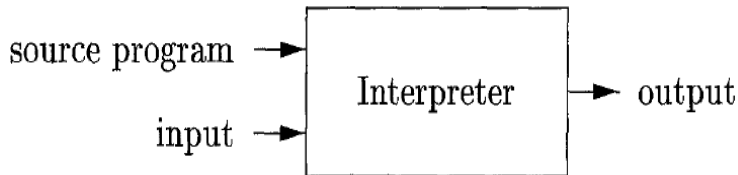
- به بیان ساده، کامپایلر برنامه‌ایست که می‌تواند یک برنامه را با یک زبان (*source language*) بخواند، و معادل آن را به زبانی دیگر ترجمه کند (*target language*).



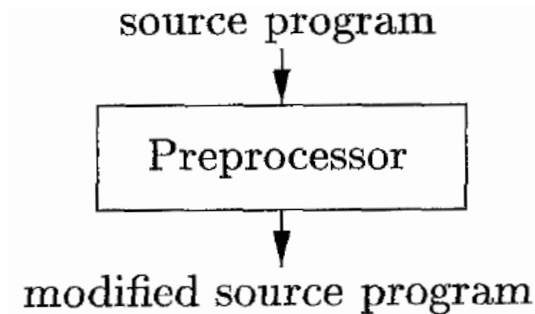
- اگر برنامه‌ی تولید شده، یک executable machine-language program باشد، می‌توان آن را مستقیماً توسط پردازنده اجرا کرد.



- نوع دیگری از پردازشگرهای زبانی، مفسرها هستند که بجای تبدیل زبان به زبان دیگر (*target*)، خود مستقیماً مسئول اجرای زبان اول (*source*) می‌شوند.



یک سیستم پردازشگر زبان - Preprocessor



- زبان‌های کهنی مثل C و C++ سیستم moduling و ساختاربندی منظمی برای جدا کردن source code هایشان نداشتند.

- اما زبان‌های جدیدتر مثل Python چنین سیستمی را دارند.

```
1 import math
2 from csv import reader, writer
```

- زبان‌های کهنی مثل C و C++ سیستم moduling و ساختاربندی منظمی برای جدا کردن source code هایشان نداشتند.
- اما زبان‌های جدیدتر مثل Python چنین سیستمی را دارند.

```
1 import math
2 from csv import reader, writer
```

- به همین دلیل، آنها نیاز داشتند که وقتی برنامه‌ای که نوشته‌اند را به چندین فایل تقسیم کرده‌اند، کامپایل کنند، برنامه‌ای تمام source code‌هایشان را به یک source code واحد تبدیل کرده و آن را به کامپایلر بدهند، که بخاطر این نیاز، نرم‌افزاری به نام Preprocessor نوشته شد.

main

cpython / Python / ceval.c

markshannon GH-111485: Generate instruction and uop metadata (GH-113287)

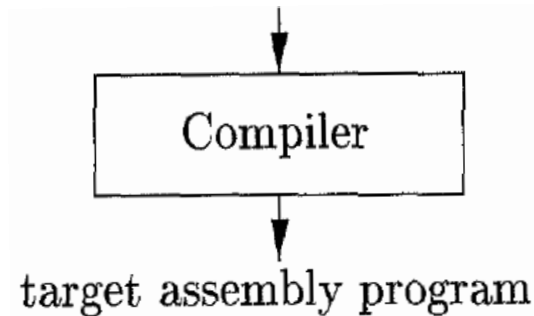
Code Blame 2903 lines (2644 loc) · 87.6 KB

```
1  /* Execute compiled code */
2
3  #define _PY_INTERPRETER
4
5  #include "Python.h"
6  #include "pycore_abstract.h"    // _PyIndex_Check()
7  #include "pycore_call.h"       // _PyObject_CallNoArgs()
8  #include "pycore_ceval.h"      // _PyEval_SignalAsyncExc()
9  #include "pycore_code.h"
10 #include "pycore_emscripten_signal.h" // _Py_CHECK_EMSCRIPTEN_SIGNALS
11 #include "pycore_function.h"
12 #include "pycore_instruments.h"
13 #include "pycore_intrinsics.h"
14 #include "pycore_long.h"        // _PyLong_GetZero()
```

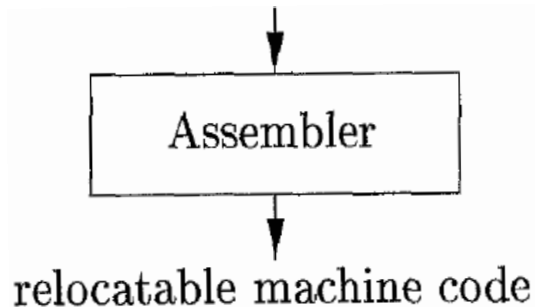
- با Preprocessor ها می توان ماکرو به زبان اضافه کرد.

```
73  #define Py_XDECREF(arg) \  
74      do { \  
75          PyObject *xop = _PyObject_CAST(arg); \  
76          if (xop != NULL) { \  
77              Py_DECREF(xop); \  
78          } \  
79      } while (0)
```

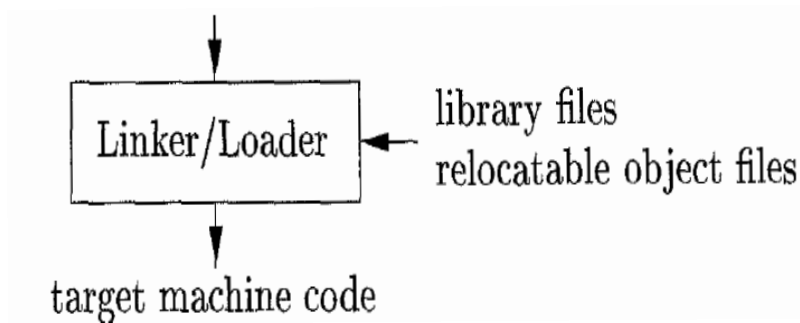
یک سیستم پردازشگر زبان - Compiler



یک سیستم پردازشگر زبان - Assembler



یک سیستم پردازشگر زبان - Linker/Loader



ساختار یک کامپایلر

- تا کنون به کامپایلر به عنوان یک جعبه دارای ورودی خروجی نگاه می کردیم،
- اما اگر این جعبه را باز کنیم، با دو قسمت اصلی در کامپایلرها مواجه می شویم:
 - ۱. آنالیز
 - ۲. سنتز

- تا کنون به کامپایلر به عنوان یک جعبه دارای ورودی خروجی نگاه می کردیم،
- اما اگر این جعبه را باز کنیم، با دو قسمت اصلی در کامپایلرها مواجه می شویم:
 ۱. آنالیز
 ۲. سنتز