به نام خدا

فاز اول پروژه درس «اصول طراحی کامپایلر»

نیم سال دوم سال تحصیلی ۱۴۰۳-۱۴۰۲

دانشکده مهندسی و علوم کامپیوتر تاریخ تحویل ۱۴۰۳/۲/۲

در این پروژه قصد داریم یک سری دستورات ساده و مرکب[1] برای یک زبان برنامه‌نویسی فرضی طراحی کنیم و کامپایلر آن را با استفاده از زیرساخت LLVM بسازیم. در نهایت object code آن را به کمک LLVM نمایش می‌دهیم.

قبل از آن که به شرح دستورات تعریف شده در زبان بپردازیم، عبارت‌های ریاضی و منطقی مجاز در این زبان برنامه نویسی را شرح می‌دهیم:

به کمک عملگرهای محاسباتی + ، - ، * ، / ، % ، ^ و () می‌توان عبارت‌های ریاضی متنوعی ساخت.
به کمک عملگرهای رابطه‌ای > ، < ، =< ، >= ، == ، =! و عملگرهای منطقی *and* و *or* و با استفاده از () می‌توان عبارت‌های منطقی متنوعی ساخت.

زبان برنامه‌نویسی فرضی ما دارای دستورات زیر است:

دستورات ساده

۱. تعریف متغیر:

در زبان طراحی شده، متغیرها دارای مقادیری هستند که در زمان compile مشخص خواهند شد. data type های موجود *int* و *bool* می‌باشند که مقداردهی اولیه ندارند. برای تعریف این دو نوع متغیر، از سینتکس‌های زیر استفاده می‌کنیم:

```
int x;
bool y;
```

می‌توان چند متغیر از یک تایپ را با هم تعریف کرد:

```
bool t = true, f = false, d;
```

[1] Simple and compound statements

می‌توان در هنگام تعریف کردن به آنها مقدار اولیه نیز داد:

```
int a = 0, b = -1, c;
```

برای نام‌گذاری متغیرها تنها مجاز به استفاده از اعداد و حروف هستیم. همچنین نام متغیر نمی‌تواند با عدد شروع شود. دقت شود کلیدواژه‌های موجود در زبان نمی‌تواند به عنوان نام یک متغیر به کار روند.

۲. انتساب متغیر:

عملگرهای انتساب =- ، =+ ، =* ، =/ و = می‌توانند برای مقداردهی متغیر *int* به کار روند اما برای متغیرهای *bool* تنها مجاز به استفاده از = هستیم.

مقادیر مجاز متغیر *bool*: *true* ، *false* و حاصل عبارت‌های منطقی معتبر

مقادیر مجاز متغیر *int*: اعداد و حاصل عبارت‌های ریاضی معتبر. توجه شود که اعداد می‌توانند با علامت مثبت، منفی و یا بدون علامت باشند. (۱+ ، ۱- ، ۱ هر سه تعریف می‌شوند) اما متغیرها را نمی‌توانیم به صورت مستقیم با علامت - قرینه کنیم. برای قرینه کردن متغیرها و عبارت‌ها می‌توان پیش از پرانتز علامت - را قرار داد. برای مثال:

```
x = -(y) - 2 * -(z / 2) + 1;
```

در این مثال قبل از عدد 2 عمل تفریق، و قبل از پرانتزها عمل قرینه کردن را داریم.

مثال‌های دیگر:

```
x = -12 * (y + z^2) / 6;
is_empty = false or ( 10 < x );
```

۳. عملگرهای یونری:

عملگرهای یونری بر روی متغیرهای نوع *int* قابل اجرا هستند:

```
x--;
x++;
```

۴. چاپ متغیر:

سینتکس این دستور به شکل `print(variable_name);` می‌باشد و تنها می‌توان مقدار یک متغیر را در کنسول چاپ کرد.

۵. کامنت:

در میان دستورات امکان کامنت گذاشتن وجود دارد. کامنت‌ها می‌توانند تک‌خطی و یا در چند خط باشند. سینتکس هردوی آن‌ها به شکل زیر است:

```
/* Comment */


/*  Comment1
    Comment2
    Comment3 */
```

دستورات پیچیده

این دستورات شامل شرط و حلقه می‌باشند. برای سادگی در بدنه آن‌ها مجاز به تعریف متغیر جدید نیستیم، اما می‌توانیم آن‌ها را به صورت تودرتو استفاده کنیم.

۱. شرط:

- if

بلوک شرط با کلیدواژه *if* آغاز می‌شود. سپس یک عبارت منطقی درون پرانتز نوشته شده و در ادامه آن، بدنه شرط در میان { } قرار می‌گیرد. در صورتی که حاصل عبارت منطقی *true* باشد، بدنه شرط اجرا می‌شود و در غیر اینصورت بدنه اجرا نمی‌شود. می‌توان به جای عبارت منطقی از یک متغیر *bool* نیز استفاده کرد.

پس از *if* می‌توانیم *else if* یا *else* نیز داشته باشیم که سینتکس آن‌ها به شکل زیر است:

```
if ( x > 10 ) {
      y = x;
}
else if ( 5 < x and x < 10 ) {
      y = 2 * x;
}
else {
     y = x / 2;
}
```

۲. حلقه:

- while

بلوک حلقه با کلیدواژه *while* آغاز می‌شود. سپس یک عبارت منطقی درون پرانتز نوشته شده و در ادامه آن، بدنه حلقه در میان { } قرار می‌گیرد. تا زمانی که شرط حلقه برقرار باشد، بدنه حلقه اجرا می‌شود. در غیر اینصورت دستور بعد از آن اجرا می‌شود. می‌توان به جای عبارت منطقی از یک متغیر *bool* نیز استفاده کرد.

مثال:

```
while ( x > 0 ) {
     y += x;
     x -= 1;
}

c = true;
while ( c ) {
     y += x;
     if ( y > 21 ) {
         c = false;
     }
}
```

- for

سینتکس این حلقه به شکل زیر است:

```
int  i;
for  ( i = 0 ; i < n ; i += 2 ) { }
for  ( i = n ; i >= 0 ; i-- ) { }
```

سه بخش داخل پرانتز با ; از هم جدا می‌شوند. در بخش اول تنها می‌توانیم مقدار اولیه متغیر *for* را مشخص کنیم و آن متغیر باید از قبل تعریف شده باشد. در بخش دوم یک عبارت منطقی قرار دارد و تا زمانی که حاصل آن *true* باشد بدنه حلقه اجرا می‌شود. در بخش سوم نیز مقدار متغیر *for* با گام‌های مشخصی عوض می‌شود. برای سادگی بیشتر، در بخش اول (i=0) متغیر iterator را تعریف نمی‌کنیم و صرفا یک متغیر که از قبل تعریف شده را مقداردهی می‌کنیم.

کدی که توسط کامپایلر شما کامپایل می‌شود علاوه بر syntax صحیح، باید semantic درستی نیز داشته باشد. بنابراین به نکات زیر توجه داشته باشید:

- قبل از آن که متغیری مقدار بگیرد، باید تعریف شده باشد.
- تعریف متغیری که قبلا تعریف شده است مجاز نیست.
- متغیرهایی که در عملیات ریاضی استفاده می‌شوند نباید از نوع *bool* باشند.
- متغیرهایی از یک نوع را نمی‌توان داخل متغیرهایی از نوع دیگر ریخت.
- تقسیم بر صفر مجاز نیست و باید تشخیص داده شود.

نکات مهم:

- از زبان C یا C++ برای توسعه کدها استفاده کنید.
- توسعه هر بخش در فایل‌های جداگانه انجام شود که قابلیت ارزیابی مجزا را داشته باشد.
- کدها خوانایی مناسبی داشته باشند و پیشنهاد می‌شود به درستی کامنت‌گذاری شود.
- کدهای خروجی به همراه گزارش و نمونه خروجی‌های اجرا شده، در یک ریپوی GitHub با دسترسی خصوصی قرار داده شود و لینک آن به عنوان خروجی، ارسال شود. همچنین لازم است حساب کاربری alidoostnia به عنوان همکار به ریپوی Github افزوده شود.
- پروژه به صورت تیمی قابل انجام است. اندازه تیم‌ها بین ۲ الی ۳ نفر قابل قبول است و در فاز بعدی، امکان تغییر اعضای تیم وجود ندارد.
- همه اعضای تیم باید در انجام پروژه مشارکت داشته باشند و تسلط هر فرد جداگانه ارزیابی خواهد شد.
- جهت پیاده‌سازی درست و کامل پروژه، پیشنهاد می‌شود اسناد مرتبط با سایت LLVM با دقت مطالعه شده و همچنین سه فصل اول کتاب learn llvm 12 مطالعه شود. کدهای نمونه و روش ارائه مطالب در کتاب به درک شما از زیرساخت یک کامپایلر مینیمال کمک خواهدکرد.
- بخش قابل توجهی از نمره‌ی پروژه شما، اجرای تست‌هایی است که در زمان ارائه به شما داده خواهد شد. صحت عملکرد هر بخش از پروژه نیز به صورت جداگانه بررسی می‌شود و در صورتی که در بخشی از پروژه اشکال داشته باشید، نمره‌ی باقی قسمت‌ها را دریافت می‌کنید.

موفق باشید