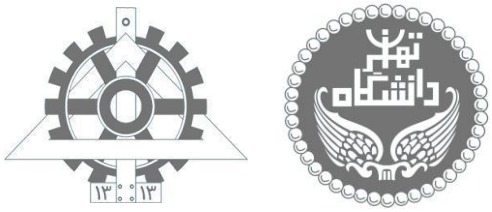

به نام خدا

فاز چهارم پروژه کامپایلرها و زبان‌های برنامه‌نویسی

Code Generation

پاییز ۱۴۰۲

مهلت تحویل: ۱۴ بهمن‌ماه۱۴۰۲

در این فاز بخش‌های مربوط به تولید کد را به کامپایلر خود اضافه می‌کنید. در انتهای این فاز، کامپایلر شما به طور کامل پیاده‌سازی شده و برنامه‌های نوشته شده به زبان UTL را به کد قابل اجرا توسط ماشین تبدیل می‌کند. پیاده‌سازی شما باید به ازای هرفایل ورودی به زبان UTL بایت کد معادل آن را تولید کند. در تست‌های این فاز، صرفا قابلیت تولید کد کامپایلرتان سنجیده می شود و ورودی‌های دارای خطاهای نحوی و معنایی که در فازهای قبل بررسی کردید نیستند؛ اما توجه کنید که شما برای تولید کد به اطلاعات جمع‌آوری شده در جدول علائم و اطلاعات مربوط به تایپ نودهای درخت AST نیاز دارید.

اسمبلر

جهت تولید فایل های class. نهایی از شما انتظار نمی‌رود که فایل باینری را مستقیما تولید کنید. برای این کار می توانید از اسمبلر jasmin که به شما معرفی شده است استفاده کنید. (در پوشه utilities تمامی موارد لازم برای تولید وجود دارد)

تساوی اشیاء

برای تایپ های int و Boolean آن‌ها را با استفاده از مقادیرشان با دستور if_icmpeq مقایسه می کنیم و برای تایپ های دیگر از دستور if_acmpeq برای مقایسه استفاده می کنیم.

عملگرهای && و ||

شما باید این عملیات را به صورت short-circuit پیاده‌سازی کنید.

نکات کلی پیاده‌سازی

- در این فاز بسیاری از مواردی که در فازها قبلی پیاده‌سازی کردید نیازی به پیاده‌سازی ندارند همچون:

- دستور و عملگر schedule را نداریم
- تنها Onstartها به ترتیبی که در کد آمده‌اند اجرا شوند
- ساختار RefreshRate نداریم.
- در میان ساختارهای حلقه تنها ساختار Whileرا داریم
- Break و Continue را نیز نداریم.
- Try و Catch را نداریم.
- Observe تنها یک trade در صورت موفقیت آمیز بودن برمی‌گرداند و در غیر این‌صورتnull برمی‌گرداند.
- در کل به تعداد ۳ عدد trade بیشتر نداریم.

- برای ساختار Connect بایستی که فرض کنید که همواره این دستور به صورت موفقیت‌آمیز عمل می کند.
- برای توسعه کد خود بایستی که از کلاس Program، Environment که در اختیارتان قرار گرفته‌است استفاده کنید. توضیحات دقیق و کافی درباره فایل Prog.java که در اختیارتان قرارگرفته است در ویدیو پیوست ارائه شده است. لطفا این ویدیو را به دقت مشاهده کنید.
- نوع بازگشتی ویزیتورهای CodeGenerator از نوع String قرارداده شده است. می توانید در هر ویزیتور یا command های تولید شده توسط آن ویزیتور را مستقیما با addCommand در فایل اضافه کنید یا اینکه مجموعه command ها را که به صورت string هستند و با \n جدا شده‌اند return کنید و درتابع دیگری آنها را به فایل اضافه کنید. پیشنهاد می‌شود ویزیتورهای expression

مجموعه commandهایشان را return کنند و دیگر ویزیتورها با گرفتن آن commandها آنها را در فایل اضافه کنند.

- در متدها و توابع، نوع تمام تایپ‌های primitive مانند intیا bool یا string را از نوع‌های غیر primitive جاوا تولید کنید. یعنی در بایت کد تولید شده باید این متغیرها از نوع‌های Integer یا Boolean یا String باشند که در java/lang هستند.
- برای Boolean ها در استک، اگر true باشد ۱ و اگر false باشد ۰ اضافه کنید.
- برای اضافه کردن مقادیر primitive به استک، از دستور ldc استفاده کنید. برای string quotation (") آن را هم دردستور ldc بیاورید.
- برای انجام محاسبات مانند add یا or روی Integer یا Boolean باید نوع آنها از نوع primitive این دو تایپ یعنی int و bool باشند. پس در تمام expression ها از نوع primitive این دوتایپ استفاده کنید و در هنگام نوشتن آنها در یک متغیر یا پاس دادن به توابع یا ریترن شدن آنها، این دو تایپ را از primitive به غیر primitive تغییر دهید. هم چنین بعد از خواندن این دو نوع از متغیر یا لیست باید تبدیل انجام شود. دلیل تبدیل‌ها آن است که در تعریف، متغیرها از نوع غیر primitive تعریف شده‌اند و در expressionها ما نیاز به primitive داریم.
- طول stack و locals را در متدها ۱۲۸ قرار دهید.
- در اضافه کردن commandها حواستان به \n ها باشد تا commandها پشت هم در فایل jasmin نباشند. هم‌چنین هر command ای که اضافه می‌کنید به طور دقیق بررسی کنید که چه آرگومان-هایی لازم دارد و چه چیزی ریترن می کند؛ زیر اگر اشتباهی رخ دهد دیباگ کردن آن در فایل‌های jasmin کار دشواری است. برای راحت‌تر دیباگ کردن و فهمیدن فایل‌های jasmin بهتر است آنها را مرتب بسازید؛ مثلا بین متدها یک خط خالی بگذارید تا مشخص باشند.

- برای مشاهده مجموعه دستورات باید کد به این لینک می‌توانید مراجعه کنید.

نکات ویزیتور های و توابع

Program

همه کلاس‌های ویزیت شوند. Current class را در CodeGenerator و Expression Type Checker ست کنید.

MethodDeclaration

Headerهای مربوطه را اضافه کنید.

VarDeclaration

دستورات مربوط به initializeکردن متغیر با توجه به تایپ آن اضافه می‌شوند.

AssignmentStmt

در این قسمت میتوانید از روی statement assignment یک node از جنسexpression assignment ساخته و آن را ویزیت کنید. توجه داشته باشید که باید در انتهای ویزیت مقداری که expression assignment روی stack قرار میدهد را pop کنید.

BlockStmt

تمام statement ها را ویزیت کنید.

ConditionalStmt

دستورات مورد نیاز برای یک شرط را اضافه کنید.

MethodCallStmt

میتوانید methodCall داخل آن را ویزیت کنید و خروجی آن را pop کنید. توجه داشته باشید که قبل و بعد از ویزیت متد کال،setIsInMethodCallStmt.expressionTypeChecker را صدا بزنید و در ابتدا آن راtrue و در انتها آن را false کنید که هنگام استفاده از expressionTypeChecker در ویزیتورها، مشکلی پیش نیاید.

PrintStmt

توابع مورد نیاز print را اضافه کنید. با استفاده از expressionTypeChecker میتوانید تایپ آرگومان را بگیرید و از signature مناسب برای print استفاده کنید. جهت نوشتن بر روی صفحه‌ی نمایش باید از print در کتابخانه‌ی java.io.PrintStream استفاده کنید.

ReturnStmt

دستورات مربوط به ریترن را اگر void نیست اضافه کنید. توجه کنید که اگر expression جلویreturn از نوع IntType یا BoolType است، ابتدا باید از primitive به غیرprimitive تبدیل شود.

NullValue

مقدار null باید روی stack گذاشته شود.

IntValue

با ldc باید مقدار آن روی stack گذاشته شود.

BoolValue

با ldc باید مقدار آن (۰ یا ۱) روی stack گذاشته شود.

StringValue

با ldc باید مقدار آن (همراه quotation) روی stack گذاشته شود.

دستورات کاربردی Jasmin

تبدیل int به Integer

```
invokestatic java/lang/Integer/valueOf(I)Ljava/lang/Integer;
```

تبدیل bool به Boolean

```
invokestatic java/lang/Boolean/valueOf(Z)Ljava/lang/Boolean;
```

تبدیل Integer به int

```
invokevirtual java/lang/Integer/intValue()I
```

تبدیل Boolean به bool

```
invokevirtual java/lang/Boolean/booleanValue()Z
```

اضافه کردن به ArrayList

```
invokevirtual java/util/ArrayList/add(Ljava/lang/Object;)Z
```

گرفتن سایز ArrayList

```
invokevirtual java/util/ArrayList/size()I
```

تبدیل (cast) یک Object به یک کلاس

```
checkcast A
```

دستورات تبدیل و اجرای کدها

کامپایل کردن فایل java. به فایل class.

```
javac -g *.java
```

اجرای فایل class. در نهایت باید Main.class اجرا شود.

```
java Main
```

تبدیل فایل بایت کد (j.) jasmin به فایل class.

```
java -jar jasmin.jar *.j
```

تبدیل فایل class. به بایت کد جاوا (نه jasmin) که خروجی در ترمینال نمایش داده می‌شود.

```
javap -c -l A
```

تبدیل فایل class. به بایت کد jasmin که خروجی در ترمینال نمایش داده می‌شود.

```
java -jar classFileAnalyzer.jar A.class
```

تبدیل class. به کد جاوا

```
drag the .class file to intellij window
```

می توانید با استفاده از دستورات بالا برای هر کد UTL که می‌خواهید معادل jasmin آن را پیدا کنید به این صورت عمل کنید که ابتدا معادل java آن کد UTL را بنویسید. سپس آن فایل جاوا را کامپایل کنید که class. تولید شود. سپس این فایل را با classFileAnalyzer به بایت کد jasmin تبدیل کنید. فقط به این نکته توجه کنید که این classFileAnalyzer یک پروژه از github بوده و لزوما خروجی صحیح نمی دهد و باید بررسی شود (در اکثر موارد خروجی درست می‌دهد مگر چند مورد خاص).

نکات مهم:

- در این فاز شما باید کد visitor مربوط به CodeGenerator که بخشی از آن به شما داده شده را تکمیل کنید. در نهایت **تنها یک فایل CodeGenerator.java** (بدون تغییرنام) آپلود کنید.
- توجه کنید که تنها **یک نفر** از هر گروه باید پروژه را آپلود کند. در صورت عدم رعایت این موارد از شما نمره **کسر خواهد شد**.
- در صورت کشف هرگونه تقلب، نمره ۱۰۰- لحاظ می شود.
- دقت کنید که خروجی‌های شما به صورت خودکار تست می شوند.
- بهتر است سوالات خود را در فروم درس یا در گروه اسکایپ مطرح نمایید تا دوستانتان نیز از آنها استفاده کنند؛ در غیر این صورت به مسئولان پروژه ایمیل بزنید:

محمد علی زمانی matin.zamani99@ut.ac.ir
نازنین ترابی گودرزی nazanin.goudarzi@ut.ac.ir
مهدی چراغی pmahdicheraghi@ut.ac.ir