به نام خدا

فاز چهارم پروژه کامپایلرها و زبان‌های برنامه‌نویسی

بهار 1401

مهلت تحویل : 21 خرداد

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 SimpleLOOP را به کد قابل اجرا توسط ماشین تبدیل میکند. پیاده‌سازی شما باید به ازای هر فایل ورودی به زبان SimpleLOOP، بایت کدهای معادل آن را تولید کند. در تست‌های این فاز، قابلیت تولید کد کامپایلرتان سنجیده می‌شود. در این فاز، ورودی‌ها دارای خطاهای نحوی و معنایی نیستند. خطاهای نحوی و معنایی در فازهای قبلی پروژه بررسی شدند. اما توجه کنید که شما برای تولید کد به اطلاعات جمع آوری شده در جدول علائم[1] و اطلاعات مربوط به تایپ nodeهای درخت AST نیاز دارید.

## اسمبلر

جهت تولید فایلهای class.نهایی از شما انتظار نمیرود که فایل باینری را مستقیما تولید کنید.

برای این کار میتوانید از اسمبلر jasmin که در کلاس معرفی شده است استفاده کنید.

[1] Symbol Table

## تساوی اشیاء

برای تایپهای int و boolean آنها را با استفاده از مقادیرشان با دستور if-icmpeq مقایسه میکنیم و برای تایپهای دیگر از دستور if-acmpeq برای مقایسه استفاده میکنیم.

## عملگرهای && و ||

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

## نکات کلی پیاده‌سازی

- نوع بازگشتی ویزیتورهای CodeGenerator از نوع String قرار داده شده است. میتوانید در هر ویزیتور یا command های تولید شده توسط آن ویزیتور را مستقیماً با addCommand در فایل اضافه کنید یا اینکه مجموعه command ها را که به صورت string هستند و با \nجدا شده‌اند return کنید و در تابع دیگری آنها را به فایل اضافه کنید. پیشنهاد می‌شود که ویزیتورهای expression، مجموعه command هایشان را return کنند و دیگر ویزیتورها با گرفتن آن command ها آنها را در فایل اضافه کنند.
- در کلاس‌ها و متدها، نوع تمام تایپ‌های primitive مانند int یا boolean را از نوع‌های غیر primitive جاوا تولید کنید. یعنی در بایت کد تولید شده باید این متغیرها از انواع Integer یا Boolean باشند که در java/lang هستند.
- برای boolean ها در stack، اگر true باشد 1 و اگر false باشد 0 اضافه کنید.
- برای اضافه کردن مقادیر primitive به stack، از دستور ldc استفاده کنید.

- برای انجام محاسبات مانند add یا or روی Integer یا Boolean باید آنها از نوع primitive یعنی int یا boolean باشند. پس در تمام expression ها از نوع primitive این دو تایپ استفاده کنید و در هنگام نوشتن آنها در یک متغیر یا pass دادن به توابع یا return شدن آنها، این دو تایپ را از نوع primitive به نوع غیر primitive تبدیل کنید. همچنین بعد از خواندن این دو نوع از متغیر یا آرایه باید تبدیل انجام شود. دلیل تبدیل‌ها آن است که در تعریف، متغیرها از نوع غیر primitive تعریف شده‌اند و در expression ها ما نیاز به primitive داریم (توابع jasmin برای تبدیل آنها در ادامه آمده است).
- طول stack و locals را در متدها 128 قرار دهید.
- فایل‌های Array.j ،Fptr.j در اختیارتان قرار گرفته‌اند و برای کار با آرایه‌ها و Fptrها باید از این کلاس‌های آماده استفاده کنید. همچنین معادل javaی آنها نیز داده شده است تا بتوانید متدهای آنها را مشاهده کنید که چه کاری انجام میدهند. برای Fptr در هنگام دسترسی به متد یکی از این کلاس ساخته میشود و instance و نام متد در آن قرار داده می‌شود. سپس در هنگام call شدن متد باید تابع invoke از این کلاس را با آرگومانهای pass داده شده صدا بزنید. توجه داشته باشید که باید آرگومانها را در یک ArrayList ذخیره کرده و به این تابع pass دهید.
- در ابتدای هر نوع constructor ای (default constructor یا تعریف شده)، ابتدا فیلدهای آن کلاس باید initialize شوند. مقادیر را به صورت زیر در نظر بگیرید: برای int مقدار صفر، برای boolean مقدار false، برای class، آرایه و Fptr مقدار null.
- نام کلاس‌ها (مثلا در signature ها یا در هنگام cast) به صورت زیر است:

IntType ⟶ java/lang/Integer

BoolType ⟶ java/lang/Boolean

ClassType ⟶ ClassName

ArrayType ⟶ Array

FptrType ⟶ Fptr

- در اضافه کردن command ها حواستان به n\ها باشد تا command ها پشت هم در فایل jasmin نباشند. همچنین هر command ای که اضافه میکنید به طور دقیق بررسی کنید که چه آرگومان‌هایی لازم دارد و چه مقداری را return می‌کند؛ زیرا اگر اشتباهی رخ دهد debugکردن آن در فایل‌های jasmin کار دشواری است. برای راحت‌تر debug کردن و فهمیدن فایل‌های jasmin بهتر است آنها را مرتب بسازید؛ برای مثال بین متدها یک خط خالی بگذارید تا مشخص باشند.
- برای مشاهده مجموعه دستورات بایت کد به این لینک می‌توانید مراجعه کنید.

## نکات ویزیتورها و توابع

### slotOf

در این تابع برای متغیرها باید slot آنها را برگردانید. توجه کنید که slot صفر به صورت پیشفرض برای object reference است و بعد از آن باید به ترتیب آرگومانهای تابع باشند؛ در واقع باید slot ها از 1 شروع شوند. طوری این تابع را پیاده‌سازی کنید که اگر ورودی یک string خالی بود، یک slot بعد از تمام slot های مخصوص آرگومان‌ها برگرداند. این برای استفاده از یک متغیر temp در code generation استفاده می‌شود؛ یعنی یک متغیر که برای تبدیل SimpleLOOP به جاوا اضافه شده است.

### addDefaultConstructor

یک default constructor به فایل اضافه کنید. فیلدهای کلاس باید initialize بشوند و constructor مربوط به superclass آن نیز در صورت وجود صدا زده شود.

### addStaticMainMethod

یک متد static main به فایل اضافه کنید. این متد توسط جاوا شناسایی شده و ابتدا این تابع در پروژه اجرا می‌شود. در این تابع باید از کلاس Main یک instance بگیرید و constructor آن را صدا بزنید (خود این تابع در فایل کلاس Main قرار میگیرد).

### Program

ابتدا تمام متغیرهای global را ویزیت کرده و بعد همه‌ی کلاس‌ها ویزیت شوند. برای ویزیت کلاس در Code Generator و در Expression Type Checker، باید current class را set کنید. همچنین در نظر داشته باشید که متغیرهای global را به صورت یک کلاس با فیلد های استاتیک بسازید بدین صورت که هرگاه نیازی به استفاده از یکی از این متغیرها شد، با کمک getstatic بتوانید متغیر را در استک push کنید.

## ClassDeclaration

فایل کلاس متناظر را با تابع createFile بسازید و سپس header مربوط به کلاس را اضافه کنید. اگر کلاس parent نداشت، parent آن را java/lang/Object قرار دهید. سپس فیلدها را ویزیت کنید. اگر constructor دارد آن را ویزیت کنید و در غیر این صورت یک default constructor اضافه کنید. در نهایت متدها را ویزیت کنید. قبل از ویزیت کردن متد یا constructor، باید current method را در Code Generator و Expression Type Checker مشخص کنید (set کنید).

## ConstructorDeclaration

اگر constructor حداقل یک آرگومان میگیرد باید یک constructor علاوه بر default constructor به کلاس اضافه شود. یعنی تمام کلاس‌ها دقیقا یک constructor بدون آرگومان (default constructor) و حداکثر یک constructor با آرگومان خواهند داشت. اگر کلاس Main است باید یک static main method هم به کلاس اضافه شود.

## MethodDeclaration

header های مربوطه را متناسب با متد یا constructor بودن اضافه کنید. اگر constructor است، constructor مربوط به superclass آن را صدا زده و فیلدهایش را نیز initialize کنید (در صورتی که superclass نداشت، constructor کلاس Object باید صدا زده شود). سپس متغیرهای local را ویزیت کنید و initialize کنید (آرگومانها را براساس مقدار پیش‌فرض initialize کنید). سپس statement ها را ویزیت کنید. دقت کنید که اگر تابع

مقدار بازگشتی ندارد، نیاز است که حتما یک دستور return در انتهای command های متد قرار داده شود. برای فهمیدن اینکه متد مقدار بازگشتی دارد یا خیر از getDoesReturn روی methodDeclaration استفاده کنید که در فاز قبل (در بخش امتیازی) تعیین شده است.

## FieldDeclaration

دستورات مربوط به field اضافه میشوند.

## VarDeclaration

دستورات مربوط به initialize کردن متغیر با توجه به تایپ آن اضافه میشوند.

## AssignmentStmt

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

### BlockStmt

تمام statement های داخل block را ویزیت کنید.

### ConditionalStmt

دستورات مورد نیاز برای یک شرط را اضافه کنید. دقت کنید که در این بخش labelهای مناسب برای elsifها، else و after را تولید کنید و در ادامه استفاده کنید.

### ElsifStmt

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 methodCall باید expressionTypeChecker.setIsInMethodCallStmt را صدا بزنید و در ابتدا setIsInMethodCallStmt را true 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

در صورتی که نوع خروجی تابع void نیست، دستورات مربوط به return را اضافه کنید. توجه کنید که اگر expression جلوی return از نوع IntType یا BoolType است، ابتدا باید از primitive به غیر primitive تبدیل شود.

### EachStmt

باید each را به for تبدیل کنید. برای این کار باید آرایه‌ای که قرار است پیمایش شود را در استک بیاورید و همچنین نیازمند به یک متغیر temp برای پیمایش هستید. سپس معادل each را که یک for روی یک لیست است، بیابید و command های قسمت‌های مربوط به initialization متغیر حلقه، شرط حلقه (متغیر temp از طول آرایه پیمایش شونده کمتر باشد)، update (اندیس temp از آرایه گرفته شده با تابع getElement در کلاس Array و در پیمایش کننده ریخته شود؛ دقت کنید که cast کردن بعد از آن فراموش نشود و temp هم باید آپدیت شود). همچنین body حلقه را باید اضافه کنید.

## BinaryExpression

برای هر یک از عملگرها دستورات مناسب را اضافه کنید. برای assignment، ابتدا بررسی کنید اگر Array دارد assign می‌شود، بعد از مجموعه دستورات operand سمت راست، دستوراتی اضافه کنید که از این آرایه که توسط دستورات operand دوم به stack اضافه شده، یک کپی ساخته شود (با copy constructor در کلاس ). سپس با توجه به اینکه lvalue از نوع identifier یا ArrayAccessByIndex یا ObjectMemberAccess است، دستورات assignment را اضافه کنید. در حالت ObjectMemberAccess دوباره دو حالت اینکه instance آن ArrayType باشد یا ClassType داریم که باید جداگانه پیاده‌سازی شوند. توجه داشته باشید که assign باید مقدار حاصل از دستورات operand سمت راست را در نهایت در stack قرار دهد تا بتوان مقدار حاصل را در slot مربوط به عملوند سمت چپ ذخیره کرد.

## UnaryExpression

برای هر یک از عملگرها دستورات مناسب باید اضافه شوند. توجه کنید که برای postdec و postinc دوباره همان تقسیم‌بندی‌های assignment وجود دارد؛ زیرا این دستورات یک مقدار جدید به آن متغیر میدهند.

## TernaryExpression

در این بخش با کمک ifeq و goto می‌توانید یک if else را پیاده‌سازی کنید.

## RangeExpression

در این بخش باید یک آرایه با کمک یک range constructor که دو ورودی leftVal و rightVal دریافت می‌کند بسازید و به حلقه each بدهید.

## ObjectMemberAccess

بررسی می‌شود که اگر نوع instance آن کلاس است، متناسب با اینکه آن member از نوع field یا متد است کد مناسب ساخته شود.

## Identifier

از slot متناسب با آن identifier باید مقدار load شود (با aload). سپس اگر نوع آن int type یا bool type است تبدیل به primitive شود. دقت کنید در این بخش اگر متغیر global باشد، نباید از slotOf استفاده کرد؛ بلکه باید فیلد استاتیک متغیر متناسب را load کنید (با استفاده از getstatic).

## ArrayAccessByIndex

با استفاده از getElement آن اندیس مورد نظر گرفته شده و سپس به تایپ مناسب cast می‌شود و اگر int یا boolean است دوباره به primitive تبدیل می‌شود.

## MethodCall

یک ArrayList ابتدا new شده و مقادیر آرگومانها بعد از visit، به این لیست add می‌شود (با java/util/ArrayList/add) و سپس با استفاده از این لیست تابع invoke از instance صدا زده می‌شود. در نهایت خروجی آن به تایپ مناسب cast شده و در صورت boolean یا int بودن تبدیل به غیر primitive می‌شود. توجه داشته باشید آرگومانها بعد از visit شدن و قبل از اضافه شدن به ArrayList ،اگر int یا boolean هستند باید به غیر primitive تبدیل شوند.

## NewClassInstance

بعد از قرار دادن آرگومانهای constructor روی stack، آن constructor صدا زده می‌شود. توجه داشته باشید آرگومانها بعد از visit شدن، اگر int یا boolean هستند باید به غیر primitive تبدیل شوند (زیرا نوع آرگومان تابعی که صدا زده می‌شود مثال Integer است نه int).

## SelfClass

reference به خود کلاس باید روی stack قرار داده شود.

### ArrayValue

ابتدا باید یک ArrayList جدید ساخته شود و آرگومانها پس از visit شدن و تبدیل شدن به نوع غیر primitive به آن اضافه شوند. سپس با استفاده از این ArrayList یک Array ساخته شود (با استفاده از constructor اول کلاس Array).

### NullValue

مقدار null باید روی stack گذاشته شود.

### IntValue

با ldc باید مقدار آن روی stack گذاشته شود.

### BoolValue

با ldc باید مقدار آن (0 یا 1) روی stack گذاشته شود.

## دستورات کاربردی Jasmin

تبدیل int به integer

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

تبدیل Integer به int

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

تبدیل Boolean به boolean

```
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 A

```
checkcast A
```

## دستورات تبدیل و اجرای کدها

کامپایل کردن فایل java .به به فایل class.

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

اجرای فایلclass . در نهایت باید Main.class اجرا شود

```
java Main
```

تبدیل فایل بایت کد (jasmin.) j 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
```

میتوانید با استفاده از دستورات بالا برای هر کد SimpleLOOP که می‌خواهید معادل jasmin آن را پیدا کنید به این صورت عمل کنید که ابتدا معادل java آن کد 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 قدیمی از github بوده و در بعضی موارد ممکن است خروجی صحیحی ندهد و باید بررسی شود (در اکثر موارد خروجی درست می‌دهد مگر چند مورد خاص).

## نمونه خروجی

```
--------------------------Compiling--------------------------
Compilation Successful

-------------------Generating Class Files-------------------
Generated: Array.class
Generated: Fptr.class
Generated: Main.class

---------------------------Output---------------------------
true
8

Process finished with exit code 0
```

## 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 تغییر نام) آپلود کنید. توجه شود 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 هر گونه تقلب، نمره 100- لحاظ می‌شود.
- دقت کنید که خروجی‌ها به صورت خودکار تست میشوند؛ پس نحوه چاپ خروجی باید عیناً مطابق موارد ذکر شده در بالا باشد. تنها موارد خواسته شده را در فایل خروجی نمایش دهید و از قرار دادن خط‌های خالی و فاصله و ... نیز خودداری کنید.
- بهتر است سوالات خود را در فروم یا گروه درس مطرح نمایید تا دوستانتان نیز از آنها استفاده کنند؛ در غیر این صورت به مسئولان پروژه ایمیل بزنید:


کیانوش عرشی kianoosharshi@gmail.com

امیر فراهانی Amirferyg@gmail.com