


<p>مدرس: قاسم ثانی</p> <p>موعد تحویل: ۸ صبح ۱۷ تیرماه ۹۶</p>	<p>به نام خدا</p> <p>طراحی کامپایلرها</p> <p>پروژه نهایی</p> <p>(نیمسال دوم ۹۵-۹۶)</p>	 <p>دانشگاه صنعتی شریف</p> <p>دانشکده‌ی مهندسی کامپیوتر</p>
--	--	--

پروژه عملی درس شامل پیاده‌سازی یک کامپایلر برای نسخه ساده‌شده‌ی C است که توضیحات کامل آن در ادامه آمده است. توجه کنید که استفاده از کدهای موجود در مرجع درس یا سایر کتب کامپایلر، در صورت تسلط بر آن کد و اعلام مأخذ اشکالی ندارد ولی استفاده از کدها و برنامه‌های موجود در سایت‌ها و کدهای سایر گروه‌ها (در همین نیمسال یا سال‌های گذشته) به هیچ وجه مجاز نیست و در اکثر موارد سبب مردودی در درس خواهد شد. در این مورد تفاوتی میان گروه دهنده یا گیرنده کد وجود ندارد.

مشخصات کامپایلر

- کامپایلر تک‌گذره است و از ۵ بخش تشکیل شده است:

بخش	بارم	توضیح
تحلیل‌گر لغوی ^۱	۰.۵	
تحلیل‌گر نحوی ^۲	۱	به روش SLR(1) (روش‌های دیگر پارس هیچ نمره‌ای نخواهند داشت).
خطا پردازی		به روش panic mode
تحلیل‌گر معنایی	۰.۵	
مولد کد میانی	۲	در قالب کدهای ۳ آدرس (در غیر این صورت نمره‌ای نخواهد داشت).

^۱ Scanner

^۲ Parser

- اخذ نمره هر بخش، منوط به پیاده‌سازی بخش‌های قبل از آن است.
- ورودی کامپایلر یک متن حاوی برنامه‌ای است که کامپایلر شما باید آن را ترجمه کند.
- خروجی کامپایلر شما یک متن حاوی کد میانی تولید شده است.
- کامپایلر شما نیاز نیست از فراخوانی‌های بازگشتی پشتیبانی کند.
- اگر پارامتر ورودی از نوع int باشد، ارسال آن از طریق مقدار^۳ است و اگر پارامتر ورودی از نوع آرایه باشد، ارسال از طریق ارجاع^۴ است.
- در نسخه‌ی ساده‌شده‌ی C، فرض کنید که هیچ اشاره‌ی رو به جلویی رخ نخواهد داد و متغیرها و توابع، قبل از استفاده تعریف می‌شوند؛ چرا که در این صورت کامپایلر تک‌گذره نخواهد شد.
- آخرین تعریف^۵ در هر برنامه، تعریف تابع main است که اجرای برنامه از آن شروع می‌شود و پروتوتایپ^۶ آن به صورت زیر است:

```
void main(void);
```

- فرض کنید تابع output، تابع از پیش تعریف شده این زبان است که مقدار پارامتر ورودی را در خروجی استاندارد چاپ می‌کند. پروتوتایپ این تابع به صورت زیر است:

```
void output(int x);
```

³ Call by value

⁴ Call by reference

⁵ Declaration

⁶ Prototype

گرامر C ساده شده

توجه کنید که پایانه‌ها پررنگ‌تر از غیرپایانه‌ها نمایش داده شده‌اند.

1. Program \rightarrow DeclarationList **EOF**
2. DeclarationList \rightarrow DeclarationList Declaration | Declaration
3. Declaration \rightarrow VarDeclaration | FunDeclaration
4. VarDeclaration \rightarrow TypeSpecifier **ID** ; | TypeSpecifier **ID** [**NUM**] ;
5. TypeSpecifier \rightarrow **int**
6. FunDeclaration \rightarrow FunReturnType **ID** (Params) CompoundStmt
7. FunReturnType \rightarrow **int** | **void**
8. Params \rightarrow ParamList | **void**
9. ParamList \rightarrow ParamList , Param | Param
10. Param \rightarrow TypeSpecifier **ID** | TypeSpecifier **ID** []
11. CompoundStmt \rightarrow { LocalDeclarations StatementList }
12. LocalDeclarations \rightarrow LocalDeclarations VarDeclaration | ϵ
13. StatementList \rightarrow StatementList Statement | ϵ
14. Statement \rightarrow ExpressionStmt | CompoundStmt | SelectionStmt |
IterationStmt | ReturnStmt
15. ExpressionStmt \rightarrow Var = Expression ; | ;
16. SelectionStmt \rightarrow **if** (GenExpression) Statement | **if** (GenExpression) Statement **else**
Statement
17. IterationStmt \rightarrow **while** (GenExpression) Statement
18. ReturnStmt \rightarrow **return** ; | **return** GenExpression ;
19. Var \rightarrow **ID** | **ID** [Expression]
20. GenExpression \rightarrow RelExpression | Expression

- 21. $\text{RelExpression} \rightarrow \text{RelExpression} \ \&\& \ \text{RelTerm} \mid \text{RelTerm}$
- 22. $\text{RelTerm} \rightarrow \text{Expression} == \text{Expression} \mid \text{Expression} < \text{Expression}$
- 23. $\text{Expression} \rightarrow \text{Expression} \ \text{AddOp} \ \text{Term} \mid \text{Term}$
- 24. $\text{AddOp} \rightarrow + \mid -$
- 25. $\text{Term} \rightarrow \text{Term} \ \text{MulOp} \ \text{Factor} \mid \text{Factor}$
- 26. $\text{MulOp} \rightarrow * \mid /$
- 27. $\text{Factor} \rightarrow (\ \text{Expression} \) \mid \text{Var} \mid \text{Call} \mid \mathbf{NUM}$
- 28. $\text{Call} \rightarrow \mathbf{ID} \ (\ \text{args} \)$
- 29. $\text{Args} \rightarrow \text{ArgList} \mid \varepsilon$
- 30. $\text{ArgList} \rightarrow \text{ArgList} \ , \ \text{Expression} \mid \text{Expression}$

قالب کد میانی

توضیح	قالب کد سه آدرس
عملوند اول و دوم جمع می‌شوند و در مقصد قرار می‌گیرند.	(ADD, S1, S2, D)
عملوند اول و دوم AND می‌شوند و در مقصد قرار می‌گیرند.	(AND, S1, S2, D)
محتوای مبدأ در مقصد قرار می‌گیرد.	(ASSIGN, S, D)
اگر S1 و S2 مساوی باشند در D مقدار true و در غیر این صورت false ذخیره می‌شود.	(EQ, S1, S2, D)
حاصل S بررسی می‌شود و در صورتی که false باشد به L جهش می‌کند.	(JPF, S, L)
پرش به L انجام می‌شود.	(JP, L)
اگر S1 از S2 کوچکتر باشد مقدار D برابر true و در غیر این صورت مقدار false می‌گیرد.	(LT, S1, S2, D)
عملوند اول در عملوند دوم ضرب می‌شود و در مقصد قرار می‌گیرد.	(MULT, S1, S2, D)
عملوند نقیض می‌شود.	(NOT, S, D)
محتوا بر روی صفحه چاپ می‌شود.	(PRINT, D)
عملوند دوم از عملوند اول کم می‌شود و در مقصد قرار می‌گیرد.	(SUB, S1, S2, D)

- از روش‌های نشانی‌دهی^۷ مستقیم (مانند t) یا غیر مستقیم (مانند @t) و مقدار صریح (مانند #5) می‌توانید در کد میانی استفاده کنید. توجه کنید که در خروجی نهایی باید به جای t، نشانی مکان این متغیر در حافظه قرار گیرد.

- برای سادگی فرض کنید آدرس متغیرها به صورت ایستا^۸ تخصیص می‌یابد.
- میزان فضای در نظر گرفته شده برای هریک از متغیرها (مانند t) ۴ بایت است.

^۷ Addressing Modes

^۸ Static

ملاحظات لغوی

- کامنت‌ها همچون زبان C به صورت `/* Comment */` است و می‌تواند پس از هر توکنی بیاید.
- توکن NUM شامل اعداد صحیح (مثبت و منفی) است.
- کلیدواژه‌ها کلمات رزرو شده هستند و نمی‌توانند به عنوان ID ثبت شوند. در گرامر داده شده، کلیدواژه‌ها مشخص شده‌اند.
- تعریف توکن‌های ID و NUM به صورت زیر است:

$\text{letter} \leftarrow [A-Za-z]$

$\text{digit} \leftarrow [0-9]$

$\text{ID} \leftarrow \text{letter} (\text{letter} \mid \text{digit})^*$

$\text{NUM} \leftarrow (+ \mid - \mid \varepsilon) (\text{digit})^+$

انجام پروژه

- پروژه باید به صورت انفرادی و یا گروه‌های دو نفره انجام شود.
- در صورتی که هرگونه سوالی در رابطه با تعریف پروژه دارید، آن را در Quera بپرسید.

با آرزوی موفقیت

قاسم‌ثانی