بسمه تعالی

طراحی کامپایلرها

نیم‌سال اول ۹۷-۹۶
مدرس: دکتر قاسم‌ثانی
دستیار آموزشی: مریم غلامعلی‌تبار

دانشکده مهندسی کامپیوتر

پروژه پایانی

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 جزء تشکیل شده است:

بخش	اجزاء	نمره	توضیح
۱	تحلیل‌گر لغوی (اسکنر)	۰٫۵	
۲	تحلیل‌گر نحوی (پارسر)	۰٫۷۵	به روش SLR(۱) – در صورت استفاده از روش‌های دیگر، هیچ نمره‌ای به پروژه تعلق نمی‌گیرد.
	خطا پرداز		به روش panic mode
۳	تحلیل‌گر معنایی	۰٫۷۵	
	مولد کد میانی	۲	در قالب کدهای ۳ آدرسه (در غیر این صورت نمره‌ای نخواهد داشت.)

- اخذ نمره‌ی 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.
- پشتیبانی از فراخوانی‌های بازگشتی نمره‌ی اضافی داشته و اجباری نیست. توجه کنید که نمره‌ی اضافی به پروژه‌ای تعلق می‌گیرد که در بخش اصلی و اجباری بدون نقص باشد.
- برای تولید جدول پارس می‌توانید از برنامه‌ی JFLAP که در کوئرا بارگذاری شده استفاده کنید. (توجه کنید که تنها برای تولید جدول پارس می‌توانید از این برنامه‌استفاده کنید، نه پارسر.)

گرامر C-Minus

توجه کنید که پایانه‌ها پررنگ‌تر از غیرپایانه‌ها نمایش داده شده‌اند.

1. program → declaration-list **EOF**
2. declaration-list → declaration-list declaration | declaration
3. declaration → var-declaration | fun-declaration
4. var-declaration → type-specifier **ID** **;** | type-specifier **ID** **[** **NUM** **]** **;**
5. type-specifier → **int** | **void**
6. fun-declaration → type-specifier **ID** **(** params **)** compound-stmt
7. params → param-list | **void**
8. param-list → param-list **,** param | param
9. param → type-specifier **ID** | type-specifier **ID** **[** **]**
10. compound-stms → **{** local-declarations statement-list **}**
11. local-declarations → local-declarations var-declaration | ϵ
12. statement-list → statement-list statement | ϵ
13. statement → expression-stmt | compound-stmt | selection-stmt | iteration-stmt | return-stmt
14. expression-stmt → expression **;** | **;**
15. selection-stmt → **if** **(** expression **)** statement **else** statement
16. iteration-stmt → **while** **(** expression **)** statement
17. return-stmt → **return** **;** | **return** expression **;**
18. expression → var **=** expression | simple-expression
19. var → **ID** | **ID** **[** expression **]**
20. simple-expression → additive-expression relop additive-expression | additive-expression
21. relop → **<** | **==**

22. additive-expression → additive-expressio addop term | term

23. addop → **+** | **-**

24. term → term ***** factor | factor

25. factor → **(** expression **)** | var | call | **NUM**

26. call → **ID** **(** args **)**

27. args → arg-list | ϵ

28. arg-list → arg-list **,** expression | expression

فهرست دستورالعمل‌های سه آدرسه قابل استفاده برای تولید کد میانی

قالب کد سه‌آدرسه	توضیح
(ADD, S1, S2, D)	عملوندهای اول و دوم جمع می‌شوند و حاصل در D قرار می‌گیرد.
(SUB, S1, S2, D)	عملوند دوم از عملوند اول کم می‌شود و حاصل در D قرار می‌گیرد.
(AND, S1, S2, D)	عملوندهای اول و دوم AND می‌شوند و حاصل در D قرار می‌گیرد.
(ASSIGN, S, D,)	محتوای S در D قرار می‌گیرد.
(EQ, S1, S2, D)	اگر S1 و S2 مساوی باشند، در D مقدار true و در غیر این صورت، مقدار false ذخیره می‌شود.
(JPF, S, L,)	محتوای S بررسی می‌شود و در صورتی که false باشد، کنترل به L منتقل می‌شود.
(JP, L, ,)	کنترل به L منتقل می‌شود.
(LT, S1, S2, D)	اگر S1 کوچکتر از S2 باشد، در D مقدار true و در غیر این صورت، مقدار false ذخیره می‌شود.
(MULT, S1, S2, D)	عملوند اول در عملوند دوم ضرب می‌شود و حاصل در D قرار می‌گیرد.
(NOT, S, D,)	نقیض محتوای عملوند S در D قرار می‌گیرد.
(PRINT, S, ,)	محتوای S بر روی صفحه چاپ می‌شود.

- از روش‌های نشانی‌دهی [1] مستقیم (مانند t)، غیر مستقیم (مانند t@) یا 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 [2] تخصیص می‌یابد.
- میزان فضای در نظر گرفته شده برای هریک از متغیرها (مانند t) ۴ بایت است.

[1] Addressing Modes
[2] static

ملاحظات لغوی

- کامنت همچون زبان C به صورت /* Comment */ است و می‌تواند پس از هر توکنی بیاید.
- کلیدواژه‌ها[۳] رزرو شده هستند و نمی‌توانند به عنوان شناسه استفاده شوند.
- تعریف توکن‌های ID و NUM به صورت زیر است:

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

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

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

$$\text{NUM} \leftarrow (+|\text{-}|\epsilon)\ (\text{digit})^+$$

ملاحظات معنایی

- برنامه ورودی شامل تعدادی تعریف [۴] متغیر و تابع است که با هر ترتیبی می‌توانند ظاهر شوند.
- 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 در هر برنامه، تعریف تابع main است که اجرای برنامه از آن شروع می‌شود و پروتوتایپ [۵] آن به صورت زیر است:

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

- در این زبان، تنها متغیرهایی از نوع int می‌توان تعریف کرد و از void تنها در تعریف توابع استفاده می‌شود.
- اگر پارامتر ورودی از نوع int باشد، ارسال آن از طریق مقدار [۶] است و اگر پارامتر ورودی از نوع آرایه باشد، ارسال از طریق ارجاع [۷] است.
- در ساختار شرطی if (قاعده‌ی ۱۵)، اگر expression مقدار غیرصفر داشته باشد، statement اول اجرا می‌شود. در غیر این صورت، statement دوم اجرا می‌شود.
- در ساختار while (قاعده‌ی ۱۶)، تا زمانی که مقدار expression غیرصفر باشد، statement اجرا می‌شود.
- در قاعده‌ی ۲۰، اگر حاصل ارزیابی relop برابر true باشد، مقدار simple-expression برابر ۱ می‌شود. در غیر این صورت، مقدار آن صفر می‌شود.

---

[۳] Keywords
[۴] Declaration
[۵] Prototype
[۶] Call by value
[۷] Call by reference

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

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

نمونه‌ی ورودی

توجه کنید که صحت خروجی کامپایلر شما توسط برنامه‌ی مفسر کد میانی بررسی خواهد شد (برنامه‌ی مفسر کد میانی به همراه مستند راهنمای استفاده از آن، در کوئرا بارگذاری شده است). بنابراین خروجی کد شما باید کاملا مطابق با قالب فوق و راهنمای همراه این برنامه باشد. یک نمونه ورودی در ادامه آمده است:

```
/* test case */
int var1;
int array1[5];

int assign1(int a){
	a=2;
	output(a);
	return 1;
}
void main(void){
	int a;
	int b;
	a=0;
	b=1;
	var1 = 3;
	if(a < var1 - 1){
		a = a*3 + 1;
		output(assign1(a));
	 }else
		while ( a == 3  && a < b ){
			int c;
			c=45;
			output(c-b) ;
		}
}EOF
```

انجام پروژه

- پروژه باید به صورت انفرادی و یا گروه‌های دو نفره انجام شود. در صورتی که میزان مشارکت اعضای گروه‌های دو نفره با یکدیگر برابر نباشد، فردی که مشارکت کمتری داشته، نمره‌ی کمتری نسبت به دیگری می‌گیرد.
- در صورتی که هر گونه سوالی در رابطه با تعریف پروژه دارید، آن را در کوئرا مطرح نمایید.
- مهلت بارگذاری سورس کد پروژه، ساعت ۸:۰۰ روز دوشنبه ۲۱ خرداد است.
- زمان‌بندی تحویل حضوری متعاقبا اعلام خواهد شد. (توجه کنید که حضور هر دو عضو گروه‌های دو نفره در جلسه‌ی تحویل الزامی است.)

موفق باشید.