بسمه تعالی

طراحی کامپایلرها

نیم‌سال اول ۹۸ـ۹۷
مدرس: دکتر قاسم‌ثانی
دستیار آموزشی: زاوش متحده

دانشکده مهندسی کامپیوتر

پروژه پایانی

پروژه عملی درس شامل پیاده‌سازی یک کامپایلر تک گذره برای نسخه ساده‌شده‌ی C است که قابلیت وجود تابع‌های تودرتو [1] به آن افزوده شده‌است. 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 در مستندات همراه پروژه اشکالی ندارد ولی استفاده از کدها و برنامه‌های موجود در سایت‌ها و کدهای سایر گروه‌ها (در همین نیم‌سال یا سال‌های گذشته) اکیدا ممنوع است و در اکثر موارد سبب مردودی در درس خواهد شد. در این مورد تفاوتی میان گروه دهنده یا گیرنده کد وجود ندارد.

[1] nested functions

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 دیاگرام انتقال (مطابق مطالب اسلاید شماره ۵) — در صورت استفاده از روش‌های دیگر (از قبیل پارس پایین‌گرد یا روش‌های مبتنی بر جدول پارس)، 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.
- در صورت عدم وجود خروجی به صورت کد میانی، شما می‌توانید برای گرفتن نمره‌ی بخش ۱ و ۲ (اسکنر، پارسر و خطا پرداز)، یک متن حاوی درخت پارس را به صورت خروجی در نظر بگیرید.
- توجه داشته باشید که در صورت نداشتن خروجی، به پروژه نمره‌ای تعلق نخواهد گرفت.
- 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.
- به عنوان یک استثناء، استفاده از برنامه‌های موجود در اینترنت برای محاسبه‌ی مجموعه‌های First و Follow (با ذکر منبع در مستندات همراه پروژه) بلامانع است.

گرامر NC-Minus و ملاحظات نحوی

گرامری که در ادامه آمده است مربوط به بخشی از زبان C است که قابلیت وجود تابع‌های تودرتو به آن افزوده شده است. در این گرامر، پایانه ها پررنگ تر از غیرپایانه ها نمایش داده شده اند. شما بایستی این گرامر را با اعمال حذف چپ گردی صریح و فاکتورگیری اصلاح کنید، به طوری که گرامر حاصل شرایط لازم برای استفاده در پارس بالا به پایین پیشگو (Predictive) را داشته باشد. به عبارت دیگر، در روند پارس نبایستی هیچگونه نیازی به بازگشت به عقب (Backtracking) داشته باشیم. سپس گرامر اصلاح شده بایستی (مطابق نمونه موجود در اسلاید شماره ۵) به یک دیاگرام انتقال تبدیل و حتی المقدور ساده شود. در این رابطه توجه به نکات زیر ضروری است.

- در اثر ساده‌سازی دیگرام، زبان گرامر نبایستی هیچ‌گونه تغییری پیدا کند.
- در دیاگرام کلی، به ازای هر یک از غیرپایانه‌ها، یک زیر دیاگرام انتقال وجود دارد.
- هر یک از زیر دیاگرام‌ها فقط یک وضعیت نهایی دارد.
- وضعیت‌های نهایی لینک خروجی ندارند و به محض رسیدن به یک وضعیت نهایی، کنترل از زیر دیاگرام جاری به زیر دیاگرام فراخواننده باز می‌گردد.
- پارس در هنگام رسیدن به وضعیت نهایی زیر دیاگرام مربوط به علامت شروع گرامر خاتمه می‌یابد.
- در هر وضعیت با استفاده از اطلاعات مجموعه‌های First و Follow و با توجه به توکن جاری، خروجی مناسب آن وضعیت جهت پیمایش انتخاب می‌شود.
- لبه‌های دارای برچسب اپسیلون در صورتی پیمایش می‌شوند که توکن جاری عضو مجموعه Follow غیرپایانه مربوطه باشد.
- اگر در یک وضعیت شرایط پیمایش هیچ‌یک از خروجی‌ها (با توجه به اطلاعات دو بند فوق) وجود نداشته باشد، یک خطای نحوی تشخیص داده می‌شود.
- خطاهای نحوی تشخیص داده شده با پیغام مناسب گزارش شده و به روش Panic Mode (مشابه روش شرح داده شده در پارس LL(1)) اصلاح می‌شوند.
- در زمان‌های مناسب در عمل تجزیه، و هنگام پیمایش برخی از لبه‌ها، بایستی تحلیل‌گر معنایی و یا تولیدکننده کد میانی توسط پارسر فراخوانی شده که روتین معنایی لازم اجرا شود. به عبارت دیگر، این فعالیت‌ها به صورت همزمان (Pipeline) با فعالیت‌های پارس و اسکن انجام می‌شوند و کامپایلر تنها با یک گذر (Pass) همه وظایف مربوط به تحلیل لغوی، نحوی، معنایی و تولید کد میانی را انجام می‌دهد.
- علاوه بر اسلاید شماره ۵، در صفحه ۲۲۳ مرجع اصلی درس توضیح مختصری راجع به روش پارس با استفاده از دیاگرام انتقال داده شده است. همچینین فایلی به فرمت PDF به منظور توضیح بیشتر این روش، به همراه تعریف پروژه در کوئرا آپلود شده است.

1. program → declaration-list **EOF**
2. declaration-list → declaration-list declaration | ϵ
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-stmt → **{** declaration-list statement-list **}**
11. statement-list → statement-list statement | ϵ
12. statement → expression-stmt | compound-stmt | selection-stmt | iteration-stmt | return-stmt | switch-stmt
13. expression-stmt → expression **;** | **continue ;** | **break ;** | **;**
14. selection-stmt → **if** (expression) statement **else** statement
15. iteration-stmt → **while** (expression) statement
16. return-stmt → **return ;** | **return** expression **;**
17. switch-stmt → **switch** (expression) **{** case-stmts default-stmt **}**
18. case-stmts → case-stmts case-stmt | ϵ
19. case-stmt → **case NUM :** statement-list
20. default-stmt → **default :** statement-list | ϵ
21. expression → var **=** expression | simple-expression
22. var → **ID** | **ID** [expression]
23. simple-expression → additive-expression relop additive-expression | additive-expression

24. relop → < | ==

25. additive-expression → additive-expression addop term | term

26. addop → + | -

27. term → term * factor | factor

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

29. call → **ID** (args)

30. args → arg-list | ϵ

31. 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 [۲] 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 [۳] تخصیص می‌یابد (مگر اینکه بخواهید فراخوانی بازگشتی را نیز پیاده‌سازی کنید).
- میزان فضای در نظر گرفته شده برای هریک از متغیرها (مانند t) ۴ بایت است.

---

[۲] Addressing Modes
[۳] static

ملاحظات لغوی

- کامنت همچون زبان C به صورت /* Comment */ است و می‌تواند پس از هر توکنی بیاید.
- کلیدواژه‌ها[4] رزرو شده هستند و نمی‌توانند به عنوان شناسه استفاده شوند.
- تعریف توکن‌های ID و NUM به صورت زیر است:

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

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

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

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

- تشخیص مثبت یا منفی بودن، یا علامت جمع یا منها بودن کاراکترهای + و - به عهده‌ی تحلیل‌گر لغوی است و بر اساس توکن قبل از آن صورت می‌گیرد.

[4] Keywords

ملاحظات معنایی

- برنامه ورودی شامل تعدادی تعریف [5] 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 غیر این صورت کامپایلر تک‌گذره نخواهد شد.
- آخرین تعریف در هر برنامه، تعریف تابع main است که اجرای برنامه از آن شروع می‌شود و پروتوتایپ [6] 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 [7] است و اگر پارامتر ورودی از نوع آرایه باشد، ارسال از طریق ارجاع [8] است.
- در قاعده‌ی ۱۳، break مخصوص بدنه‌ی switch و while است. همچنین continue نیز تنها در بدنه‌ی while قابل استفاده است.
- در ساختار شرطی if (قاعده‌ی ۱۴)، اگر expression مقدار غیرصفر داشته باشد، statement اول اجرا می‌شود. در غیر این صورت، statement دوم اجرا می‌شود.
- در ساختار while (قاعده‌ی ۱۵)، تا زمانی که مقدار expression غیرصفر باشد، statement اجرا می‌شود.
- در ساختار switch (قاعده‌ی ۱۷)، در صورت نبود break، دستورات case های جلوتر نیز اجرا خواهند شد.
- در قاعده‌ی ۲۳، اگر حاصل ارزیابی relop برابر true باشد، مقدار simple-expression برابر ۱ می‌شود. در غیر این صورت، مقدار آن صفر می‌شود.
- 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);
```

---

[5] Declaration
[6] Prototype
[7] Call by value
[8] Call by reference

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 شده‌است). 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 function1(int a){
    void function2(int b, int c){
        if(c < 2)
            output(b);
        else
            output(a);
    }
    int d;
    int function3(void){
        int c;
        c = 0;
        switch(var1){
            case 1:
                c = a;
                break;
            case 2:
                c = a - 2;
            case 3:
                c = c + 1;
                break;
            default:
                c = 9;
        }
        return c;
    }
    d = 4;
    var1 = 2;
    function2(function3(), d);
    return 1;
}
int array1[5];
void main(void){
    int i;
    i = 5;
    if (function1(i))
        return;
    else;
    while(i){
        array1[i = i - 1] = i;
        -37;
        output(array1[i]);
    }
}
```

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 از طریق کوئرا مطرح نمایید.
- مهلت بارگذاری سورس کد پروژه، ساعت ۸:۰۰ روز یکشنبه ۷ بهمن است.
- زمان‌بندی دقیق تحویل حضوری متعاقبا اعلام خواهد شد. (توجه کنید که حضور هر دو عضو گروه‌های دو نفره در جلسه‌ی تحویل الزامی است.)

موفق باشید.