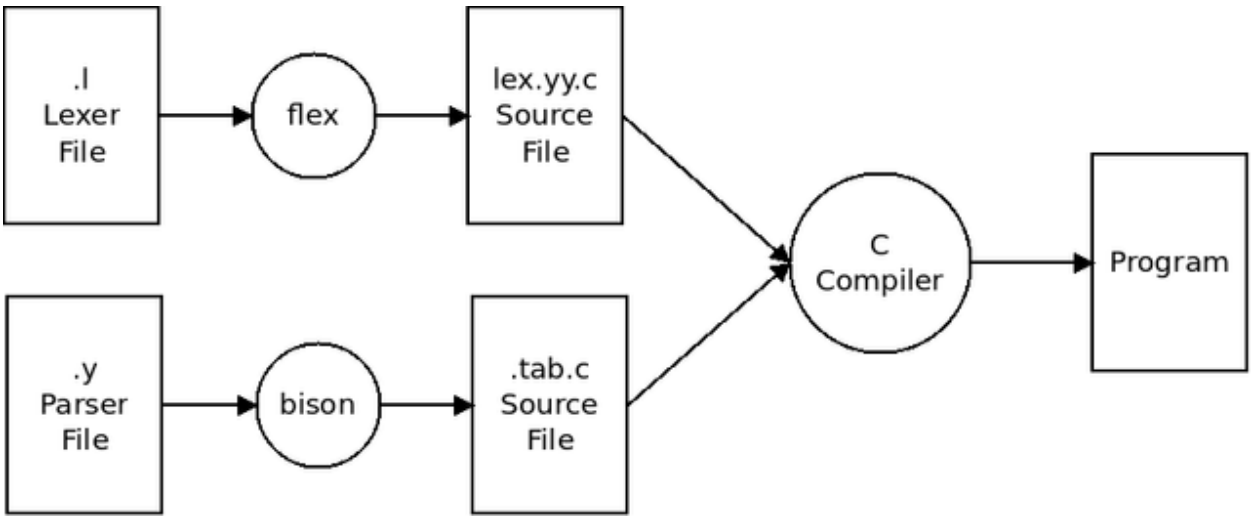
پروژه درس کامپایلر

استاد درس: دکتر فاطمه منصوری

بهار 1400

در این پروژه قصد داریم تا به کمک ابزار flex و bison کامپایلری برای یک زبان ساده طراحی کنیم و ماژول‌های اصلی آن (Lexical Analyzer, Syntax Analyzer, Semantic Analyzer, ...) را پیاده سازی کنیم، و در نهایت کدِ زبان را به کدِ اسمبلی MIPS ترجمه کنیم.



زبان ما بسیار به زبان C شباهت دارد و برای سادگی قسمت هایی از آن حذف شده است.

مثالی از زبان ما:

```
int positiveSum(int a, int b)
<
    int sum.
    sum = a + b.
    if(sum)
    <
        return sum.
    >
    else
    <
        return 0.
    >
>

int main()
<
    int a = 8.
    int b = 5.
    int counter = 0.
    while(positiveSum(a, b))
    <
        counter = counter + 1.
        a = a - 1.
        b = b - 1.
    >
    $$ end of program
    return 0.
>
```

توکن های زبان به شکل زیر است:

- کلمات کلیدی زبان

کلماتی هستند که در یک زبان برنامه نویسی نمی توان از آن ها به عنوان مفهوم دیگری مانند نام متغیر ها استفاده کرد. کلمات کلیدی مورد استفاده در این پروژه برای زبان مورد نظر ما، شامل موارد زیر می باشد:

int	char		
if	else	elseif	
while	continue	break	for
return	void	main	

- نام متغیر ها یا توابع

باید ترکیبی از حروف انگلیسی بزرگ یا کوچک، اعداد و ’_‘ باشد که نمیتواند با عدد شروع شود.

- مقادیر ثابت

میتوانیم از اعداد صحیح (مثبت یا منفی) برای متغیر های int و از کاراکتر ها برای char استفاده کنیم.

بزرگ ترین و کوچکترین مقدار برای اعداد صحیح متناسب با یک سیستم 32 بیتی میباشد.

مقدار های ثابت برای کاراکتر ها بین ' ' قرار میگیرد.

- کامنت های یک خطی و چندخطی

کامنت های تک خطی با $$ و کامنت های چندخطی با *$ شروع و با $* پایان میابند.

- عملگر ها

در این زبان عملگر های شرطی، منطقی یا محاسباتی داریم:

<	<=	==	!=	>	>=
|	&	||	&&	^	!
+	-	*	/		

- سایر توکن ها
که در هیچکدام از دسته های بالا قرار نمیگیرند

( ) [ ] < > , .

- دقت شود که هر تعداد فاصله بین دو توکن بی تاثیر است و باید نادیده گرفته شود.

قواعد زبان به شکل زیر است:

- برنامه شامل یک تابع main است.
- If
ساختار آن شبیه به زبان c است

```
if (condition)
<
	$$ code here
>
elseif (condition)
<
	$$ code here
>
else
<
	$$ code here
>
```

- while

```
while(condition)
>
	$$ code here
<
```

- for

for(variable definition. condition. step)

<

$$ code here

>

- تعریف متغیر
میتوانند به هریک از صورت های زیر تعریف شوند:

int a.

char w = ‘q’.

- توابع
توابع موجود در برنامه، پروتوتایپ ندارند و حتما قبل از تابع main تعریف میشوند.
میتوانند بدون آرگومان ورودی باشند یا حداکثر 3 آرگومان ورودی داشته باشند.
نوع خروجی آن ها هم میتواند void، int و یا char باشد.

- تمامی دستورات به جز دستورات شرطی و حلقه ها با کاراکتر . (dot) پایان میپذیرند.

- هر بلاک کد بین دو <> قرار میگیرد.

وارد بخش Semantic Analyzer میشویم:

- نکته مهم در مورد این بخش این است که ، باید بلاک هر متغیر را کنترل کنید، یعنی اگر متغیری در بلاک for تعریف میشوند، فقط در همان بلاک قابل استفاده است.
- تقسیم بر صفر را با چاپ یک هشدار کنترل کنید.
- باید semantic action ها را طوری تعریف کنید تا کد اسمبلی به زبان MIPS تولید کنند.

خطا ها:

- برنامه ی شما باید بتواند هرگونه خطا در فایل ورودی را تشخیص دهد، و آن را چاپ کند.
- در صورت وجود خطا عمل کامپایل بدون تولید کدی پایان پذیرد.

نمره اضافه:

- پیاده سازی آرایه ها و قابلیت استفاده از اعضای آن در بخش های مختلف برنامه، مانند زیر:

  int array[CONST_NUM].

  if(array[i] == 2)

  < ... >

  همینطور خطاهای مربوط به آرایه مانند outofRange را کنترل کنید.
- پیاده سازی دستورات break, continue در حلقه ها
- پیاده سازی متغیر های global قبل از تابع main
- پیاده سازی switch case به صورت زیر:

```
switch (caseSwitch)
<
    case 1:
        ...
        break.
    case 2:
        ...
        break.
    default:
        ...
        break.
>
```

شیوه تحویل:

لطفا فایل های خود را با فرمت studentNumber1_studentNumber2.zip زیپ کرده و آپلود نمایید.