طراحی کامپایلر(۴۱۴-۴۰)

سمانه حسین‌مردی

بهار ۱۴۰۲	پایان‌ترم	مدت زمان: ۳ ساعت - نمره کل: ۲۰۰

پارسر LR(1)

سوال ۱ (۳۵ نمره)

گرامر زیر را در نظر بگیرید با توجه به آن به پرسش‌های زیر پاسخ دهید:

(ϵ به معنی رشته‌ای به طول صفر می‌باشد. و علامت $ انتهای رشته را نشان می‌دهد.)

S → Ab

A → (bA)

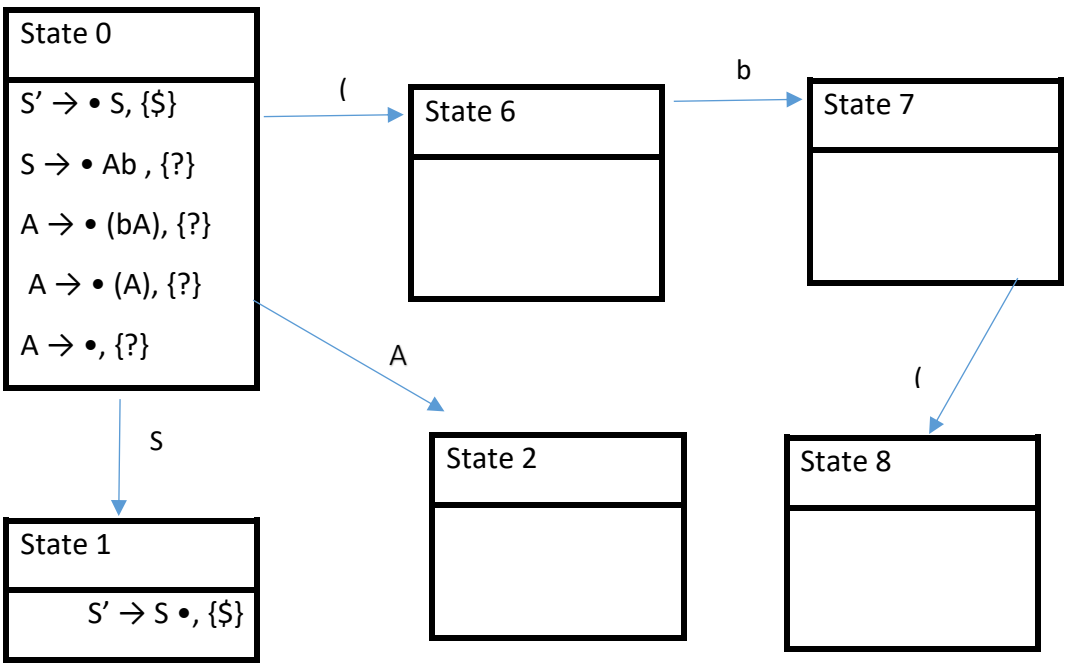
A → (A)

A → ϵ

الف- در state 0 جای علامت سوال را پرکنید. (lookAhead ها را کامل کنید)

ب - stateهای مشخص شده را کامل کنید. (در هر state می‌بایست item(1) که شامل item(0) و lookeAhead می‌باشد را مشخص کنید و در صورت نیاز Closure گیری نمایید.)

پ- جدول پارس را برای State 0 و State 1 رسم کنید. (جدول شامل دو بخش Action و Goto می‌باشد)



پارسر تقدم عملگر (operator precedence)

سوال ۲ (۳۰ نمره)

گرامر زیر را باتوجه به نکات بیان شده برای پارسرتقدم عملگر (operator precedence) در نظر بگیرید:

- ترمینال‌ها (Terminals) : {i , t , e, a ,b} و $ علامت پایان رشته می‌باشد.
- غیرترمینال‌ها (None-Terminals) : {S, C}
- (.> علامت کوچکتر) و (<. علامت بزرگتر) و (.=. علامت مساوی) در این روش پارس می‌باشند.

الف- first term و last term غیرپایانه‌ها (None-terminal) را مشخص کنید.

ب- جدول پارس تقدم عملگر (operator precedence) را بدست آورید.

پ- رشته ibtaea را با استفاده از جدول پارس تقدم عملگر (operator precedence) پارس کنید و جدول مربوط به آن (شامل Stack, input ,Action) را کامل کنید.

$S \rightarrow i\ C\ t\ S\ e\ S \mid a$

$C \rightarrow b$

تحلیل معنایی (Semantic Analysis)

سوال ۳ (۲۰ نمره)

کد برنامه زیر را درنظر بگیرید:

```
1:      int x = 0;
2:      int *y = malloc(sizeof(int));
3:      while(x < 10){
4:          int z = 0;
5:          int y = x;
6:          while(y < 10){
7:              z = z + y;
8:              y = y + 1;
9:          }
10:     }
11:     *y = x;
```

الف- جدول نمادها (Symbol Table) و Scope Stack را در زمان کامپایل برای متغیرهای کد خط‌های ٤ و ١١ را رسم نمایید.

جدول نمادها شامل موارد زیر می‌باشد:

(Lexeme) , (proc/ func/ var), (No. Arg/ Cell) , (type) , (scope)

تولید کد میانی (Code Generation for Procedures)

سوال ۴ (۲۰ نمره)

کد برنامه زیر را در نظر بگیرید.

```
int  baz(int y)  {
        return y*y;

 }

int  foo()  {
        int x;
         x = 3;
         x += baz(x);
         return x;

}
```

الف- کد میانی مربوط به آن توسط یک کامپایلر (بالا پایین) را براساس دستورات کد سه آدرسه (three-address code) بنویسید.

Run Time Memory

سوال ۵ (۲۵ نمره)

قطعه کد زیر را در نظر بگیرید:

```
void main() {
    int x;
    int y;
    float f;
     ... //some computations
    f = foo(x + y);
    f += bar(x, y);
    ... //some computations
      }
float foo(int a) {
    float g;
    g = bar(a, a);
    return g;
}
    float bar(int r, int s) {
    float h;
    h = 1.0 * (r + s);
    return h;
}
```

الف- فرض کنید برنامه با استفاده از callee saves روی ماشینی با ٤ رجیستر اجرا می شود. فرض کنید آدرس‌ها (i.e., pointers) ٨ بایت، floats ٤ بایت و int ٤ بایت است. پشته کامل (یعنی پشته شامل تمام activation recordهای فعال) را برای برنامه درست بعد از اینکه Foo تابع bar را فراخوانی کرده است و قبل از بازگشت bar رسم کنید. برای هر قسمت (slot) در پشته، مشخص کنید چه چیزی در آنجا ذخیره شده است و چه مقدار فضای آن قسمت (slot) را اشغال می کند.

ب- سپس فرض کنید که برنامه از caller saves استفاده می‌کند. پشته کامل برنامه را درست بعد از فراخوانی bar توسط Main و قبل از بازگشت bar بکشید.

Activation Record شامل موارد زیر می‌باشد:

Activation Record: (return value, actual parameters, optional control link, optional access link, saved machine, status local data, temporaries)

Optimization

سوال ۶ (۲۵ نمره)

قطعه کد زیر را در نظر بگیرید (n یک مقدار ثابت است، A یک آرایه سراسری است؛ فرض کنید که متغیرهای محلی بعد از حلقه ها استفاده نمی شوند):

```
1:      i = 0;  t0 = 0;  t1 = 0;  t2 = 0;
2:      while ( t0 < n ) {
3:              t0 = 2 * i + 1;
4:              for j = 0 to i {
5:                      t1 = 2 * t0 + j;
6:                      t2 = 7 * t1;
7:                      A[i, j] = (j * t1) + t2;
                }
8:              if (t1 < 0) {
9:                      t0 = t0 * t1;
               }
10:                     i = i + 1;
    }
```

الف- نمودار جریان کنترل (control-flow graph) این کد را با توجه به بلوک‌های پایه (Basic block) و لبه‌های (edge) بین آن‌ها رسم کنید. (در هر بلاک پایه شماره دستورالعمل ها را بنویسید)

ب- دستورات مستقل از حلقه را در جای مناسب قرار دهید

ت- بهینه سازی محلی در سطح بلوک پایه را برای کد مقابل انجام دهید.

ث- با فرض اینکه کل برنامه تولید شده کد مقابل است، آیا دستوری وجود دارد که بتوان آن را حذف کرد؟

Register allocation

سوال ۷ (۴۵ نمره)

کد زیر رادر نظربگیرید:

```
1:  A = 7
2:  B = 8
3:  C = A + B
4:  D = A + C
5:  B = C - D
6:  E = A + B
7:  A = D + C
8:  F = A + C
9:  G = E + F
10: WRITE(G) //this counts as a use of G
```

الف - بعد از هر دستور نشان دهید که کدام متغیرها live هستند.

1	
...	
10	

ب- گراف تداخل (Interface Graph) را برای آن رسم کنید.

پ- کمترین تعداد رجیستر را برای این گراف به گونه‌ای مشخص کنید که نیاز به ریختن (splling) نباشد. دلیل آن را توضیح دهید.

ت- با فرض وجود ۳ رجیستر، تخصیص رجیستر از پایین به بالا را روی این قطعه کد انجام دهید. در هر دستور، نشان دهید که پس از اجرای دستور، رجیستربه کدام متغیر نسبت داده شده است (اگر یک ثبات آزاد شد، آن را علامت گذاری کنید، حتی اگر هنوز مقداری دارد). در صورت نیاز از برای رجیسترها از ریختن (splling) استفاده کنید و محل وقوع load و store ناشی از ریختن (splling) را مشخص کنید.

Inst	R1	R2	R3	Loads/Stores due to spills
1				
...				
10				

موفق باشید ☺☺