آزمون پایانی درس طراحی کامپایلر

جمع نمره‌ها: ۱۰۰ زمان آزمون: ۸۰ دقیقه

---

۱ (۱۶) به پرسش‌های زیر دقیق و کوتاه پاسخ دهید.

۱.۱ دو روش برای تعیین اولویت (Precedence) و جهت شرکت‌پذیری (Associativity) عملگرها در تجزیه بیان کنید.

۲.۱ دو دلیل برای اصلاح خطاهای نحوی در کامپایلر ذکر کنید.

۳.۱ تفاوت بهینه‌سازی‌های محلی و سراسری چیست و کدام پیچیدگی کمتری دارد؟

۴.۱ دو عیب برای جمع‌آوری زباله (Garbage collection) با شمارش منابع (Reference counting) بیان کنید.

۲ (۳۰) حروف در گرامر زیر «cir»، «box»، «ht»، «wd» و «n» هستند. در صورت نیاز، این گرامر را تغییر دهید و سپس با الگوریتم LL(1) عمل تجزیه (Parse) را مطابق با گام‌های زیر انجام دهید:

```
S → cir A S
S → box A S
S → ε
A → ht n A
A → wd n A
A → ε
```

۱.۲ مقدار توابع Nullable، First و Follow را برای متغیرها محاسبه نمایید.

۲.۲ جدول LL(1) را بکشید.

۳.۲ با نشان دادن وضعیت پشته، رشته‌ی ورودی و عمل در هر گام، رشته‌ی «box ht n wd n cir» را با استفاده از الگوریتم LL(1) تجزیه کنید.

۳ (۳۰) کد میانی روبرو که به صورت کد سه-آدرسه (Three-address code) است، را نظر بگیرید.

```
01   dx = x2 - x1
02   dy = y2 - y1
03   if dx >= 0 goto 08
04   t1 = x1
05   x1 = x2
06   x2 = t1
07   dx = x2 - x1
08   goto 11
09   dx = dx / dy
10   dy = 1
11   if dx < 10 goto 15
12   dx = dx / c
13   dy = dy / c
14   goto 11
15   goto 17
16   if dy > 1 goto 09
17   sum = dx + dy
```

۱.۳ بلوک‌های پایه (Basic blocks) را بدست آورید.

۲.۳ گراف جریان (Flow graph) را بکشید.

۳.۳ بلوک‌های پایه‌ای که می‌توان از کد میانی حذف کرد را با دلیل مشخص کنید.

۴ (۱۴) برای کد میانی زیر، متغیرهای زنده‌ی ورودی به هر دستور و خروجی از آن را محاسبه کنید و از روی آن گراف تداخل رجیستر (Register interference) را بکشید.

```
01   c = 0
02   if a < b goto 05
03   a = a – b
04   goto 02
05   c = a * 2
06   return c
```

۵ (۱۰) درخت فعال‌سازی (Activation tree) را برای برنامه‌ی روبرو که به زبان C نوشته شده است، بکشید (اجرا با فراخوانی تابع main شروع می‌شود). سپس به این سؤال پاسخ دهید: حداکثر تعداد رکوردهای فعال‌سازی (Activation record) که همزمان روی پشته (Stack) قرار می‌گیرند، چند است؟ وضعیت پشته (ترتیب رکوردهای فعال‌سازی قرار گرفته در آن) را در آن وضعیت نمایش دهید.

```
int g(void) {
     return 4;
}

int f(int n) {
     if (n < 2)
          return g() – 3;
     return n * f(n – 1);
}

int main(void) {
     f(g());
     return 0;
}
```