تمرین‌های برنامه‌نویسی درس اصول طراحی کامپایلر

تمرین‌های اختیاری

جستجو برای میانه، کوتاه‌تر. برنامه‌ای در کد میانی تسلنگ (TSIR) پیاده‌سازی کنید که میانه‌ی یازده عدد را محاسبه نماید. تلاش کنید تعداد دستورات این برنامه کمینه شود. این تمرین تا تاریخ ۹۸/۷/۱ قابل انجام است.

کد میانی تسلنگ و دنیای واقعی. برنامه‌ای بنویسید که کد میانی تسلنگ (TSIR) را به کد اسمبلی معماری X86 یا ARM تبدیل کند. باید به ازای هر تابع در کد میانی، تابعی در کد اسمبلی ایجاد شود و رجیسترهای TSIR روی پشته قرار گیرند. این تمرین تا تاریخ ۹۸/۷/۲۲ قابل انجام است و برای سه نفر اول (یا گروه دو نفره) حدود یک نمره دارد.

دریای روابط ریاضی. برنامه‌ای بنویسید که عبارت‌های ریاضی به شکل لاتک (پیوند) را در ورودی نیتراف به شکل عبارت‌های ریاضی Eqn تبدیل کند. برای نمونه، عبارت «a^{\frac{b}{c}}» با «a sup {b over c}» جایگزین می‌شود. لازم است ابتدا نیتراف را نصب کنید (پیوند). این تمرین تا تاریخ ۹۸/۸/۱۳ قابل انجام است و برای سه نفر اول (یا گروه دو نفره) حدود یک و نیم نمره دارد. با کامل کردن و آزمایش بیشتر می‌توانید کارتان را برای درس پروژه‌ی کارشناسی (پروژه‌ی نهایی دوره‌ی کارشناسی) نیز ارائه دهید.

دسته‌ی شصت و چهارتایی برای نیت‌سی‌سی. هدف این تمرین، تولید کد ARM64 یا MIPS برای کامپایلر NEATCC (پیوند) است. برای اطلاعات بیشتر در مورد نیت‌سی‌سی، فایل معرفی آن (پیوند) را بخوانید. این تمرین تا تاریخ ۹۸/۸/۲۰ قابل انجام است و برای سه نفر اول (یا گروه دو نفره) حدود سه نمره دارد. با کامل کردن و آزمایش بیشتر می‌توانید کارتان را برای درس پروژه‌ی کارشناسی (پروژه‌ی نهایی دوره‌ی کارشناسی) نیز ارائه دهید.

کامپایلر تسلنگ: گام اول

در گام اول از تمرین عملی درس طراحی کامپایلر، باید یک تحلیلگر لغوی بنویسید. در این گام باید برنامه‌ای بنویسید که با خواندن یک فایل در زبان تسلنگ از ورودی استاندارد، واژه‌های (Tokens) آن را چاپ کند. برای مثال، کد تسلنگ زیر را در نظر بگیرید:

```
vecsum(A: list): num
{
	local n: num;
	n = (5 + 2);
	return n;
}
```

برای مثال، اگر این قطعه کد به عنوان ورودی داده شود، برنامه‌ی شما باید هر واژه را در یک خط خروجی چاپ نماید:

```
vecsum
(
A
:
list
)
:
num
{
local
n
:
num
;
n
=
(
5
+
2
)
;
return
n
;
}
```

دقت کنید که واژه‌ها ممکن است با فاصله جدا نشده باشند. این تمرین تا تاریخ ۹۸/۸/۱۱ قابل انجام است.

کامپایلر تسلنگ: گام دوم

در گام دوم از تمرین عملی درس طراحی کامپایلر، تحلیل نحوی را انجام می‌دهید. در این گام برنامه‌ای می‌نویسید که با خواندن یک فایل تسلنگ از ورودی استاندارد و تحلیل نحوی آن، پیغام‌هایی را چاپ می‌کند. دقت کنید که تجزیه‌ی برنامه‌ی ورودی الزامی است و منطق برنامه باید به کمک عملیات مفهومی نوشته شود. در این پیغام‌ها، خطاهای نحوی را گزارش کنید: انتساب مقداری با نوع ناسازگار به یک متغیر (مثل انتساب یک عدد به یک لیست)، انجام عملی ناسازگار روی دو متغیر (مثل جمع دو لیست) و فرستادن مقادیر ناسازگار به یک تابع. برای مثال، فایل زیر را در نظر بگیرید:

```
1     find(A: list, x: num): num
2     {
3           local i: num;
4           local n: num;
5           i = 0;
6           for (n in A) {
7                 if (n == x)
8                       return i;
9                 i = A + 1;
10          }
11          return -1;
12    }
13
14    main(): num
15    {
16          local A: list;
17          local a: num;
18          A = vec(3);
19          A = 5;
20          A[2] = A;
21          A[1] = 8;
22          A = find(A, 5);
23          numprint(find(a, 5));
24          return 0;
25    }
```

برنامه‌ی شما پس از خواندن این فایل باید خط‌های زیر را چاپ نماید.

```
9: incompatible operands!
19: illegal assignment!
22: illegal assignment!
23: illegal parameter!
```

این تمرین تا تاریخ ۹۸/۹/۱۸ قابل انجام است.

کامپایلر تسلنگ: گام سوم

در گام سوم از تمرین عملی درس طراحی کامپایلر، برنامه‌ای می‌نویسید که با خواندن یک فایل تسلنگ از ورودی استاندارد، کد میانی آن را تولید می‌کند. برای نمونه، کد زیر را که در زبان تسلنگ است در نظر بگیرید.

```
sum3(a: num, b: num, c: num): num
{
	return a + b + c;
}

main(): num
{
	local a: num;
	local b: num;
	local c: num;
	a = numread();
	b = numread();
	c = numread();
	numprint(sum3(a, b, c));
	return 0;
}
```

برنامه‌ی شما پس از خواندن این فایل باید کد میانی تسلنگ را تولید کند. یک خروجی نمونه برای این دو تابع در ادامه نشان داده می‌شود.

```
proc  sum3
   add   r0, r0, r1
   add   r0, r0, r2
   ret

proc  main
   call  iget, r3
   call  iget, r1
   call  iget, r2
   call  sum3, r3, r1, r2
   call  iput, r3
   mov   r0, 0
   ret
```

این تمرین تا تاریخ ۹۸/۱۰/۱۱ قابل انجام است. به نکته‌های زیر توجه کنید:

- برای بررسی درستی کد تولید شده، با استفاده از برنامه‌ی tsvm کد میانی را اجرا کنید.
- برای خواندن ورودی و خروجی می‌توانید از توابع داخلی TSIR استفاده کنید.
- فرض کنید بردارهای تسلنگ با اندازه‌ی n، یک اشاره‌گر به قسمتی از حافظه با اندازه‌ی $n + 1$ عدد هستند. عدد اول طول بردار و سایر عددها محتویات بردار را نشان می‌دهند.

- گروه‌های یک نفره لازم نیست برای بردارها کد تولید کنند و می‌توانند فرض کنند در برنامه‌ی ورودی از بردار استفاده نمی‌شود.