گام سوم تمرین عملی درس طراحی کامپایلر

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.

```
proc sum3: num a, num b, num c -> num
{
	return a + b + c;
}

proc main -> num
{
	num a, b, c;
	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
```

به نکته‌های زیر توجه کنید:

- برای بررسی درستی کد تولید شده، می‌توانید 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 $n + ۱$ 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.
- به برنامه‌ای که کوچک‌ترین کد میانی را تولید کند نمره‌ی اضافه اختصاص می‌یابد.
- به تولید کد برای تشخیص دسترسی به خانه‌های غیر مجاز یک بردار (در زمان اجرای کد میانی) نمره‌ی اضافه اختصاص می‌یابد.
- به تبدیل کد میانی تسلنگ به کد نهایی یک معماری (مثلا کد اسمبلی X86) نمره‌ی اضافه اختصاص می‌یابد.
- می‌توانید به جای تولید کد میانی تسلنگ، کد میانی llvm را تولید کنید.