به نام خدا

فاز چهارم پروژه طراحی و پیاده‌سازی کامپایلر تولید کد ماشین MIPS

پاییز و زمستان ۹۶

تحویل: ساعت ۲۳:۵۵ شنبه، ۱۶ دی

در این فاز، بخش های مربوط به تولید کد را به کامپایلر خود اضافه می‌کنید و در انتهای این فاز، کامپایلر شما به طور کامل پیاده‌سازی شده و برنامه‌های نوشته شده به زبان Atalk را به کد قابل اجرا توسط ماشین تبدیل می‌کند. کدسازی که پیاده‌سازی می‌کنید، باید برای هر فایل ورودی به زبان Atalk، دستورهای اسمبلی معادل آن را برای ماشین مقصد MIPS تولید کند. برای نمره‌دهی این فاز، کد اسمبلی تولید شده توسط کدساز شما توسط شبیه‌ساز SPIM اجرا و خروجی برنامه شما به طور خودکار بررسی می‌شود. بررسی صحت اجرای کد تولید شده‌ی شما در بسیاری از موارد به وسیله‌ی تابع write و چاپ مقادیر ثابت یا متغیرها انجام می‌شود. به همین دلیل در صورت نادرست پیاده‌سازی کردن دستورهای read و write ممکن است نمره‌ی زیادی را از دست دهید.

در تست‌های این مرحله صرفاً قابلیت تولید کد کامپایلرتان سنجیده می‌شود و ورودی‌ها دارای خطای نحوی و معنایی (که در زمان کامپایل بررسی شده‌اند) نیست. لذا نیازی به بررسی‌های معنایی فاز قبل ندارید. دقت کنید که قسمت جمع‌آوری اطلاعات، تایپ‌ها و ساخت جدول علائم را در صورت نقص می‌بایست اصلاح کنید زیرا از اطلاعات آن‌ها در تولید کد استفاده خواهد شد.

پیاده‌سازی تک‌ریسمانه[1]

در پیاده‌سازی شما از زبان Atalk نیازی به اجرای موازی اکتورها نیست و صرفاً ناهمگام[2] بودن ارتباط آن‌ها مهم است. بنابراین همه‌ی اکتورها در یک ریسمان[3] اجرا می‌شوند و تنها دستوری که ریسمان را بلاک می‌کند تابع read است. در این صورت لازم است که یک Scheduler، ترتیب اجرای اکتورها را مشخص نماید و پس از بررسی پیام یک اکتور، ریسمان را در اختیار اکتور دیگری قرار دهد. این Scheduler در پیاده‌سازی شما بسیار ساده است و کافی است ریسمان به صورت Round-Robin به اکتورها داده شود؛ بدین شکل که ریسمان به اولین اکتور (اکتوری که در کد ورودی بالاتر از بقیه تعریف شده است) داده شود و پس از بررسی پیام داخل صف (اگر پیامی در صفش باشد)، ریسمان به اکتور بعدی (اولین اکتور پایین‌تر از آن در کد ورودی) داده می‌شود و این فرآیند تا زمانی که هیچ پیامی در صف اکتورها نباشد ادامه می‌یابد. در صورتی که پیامی در صف اکتورها نباشد اجرای برنامه به پایان می‌رسد.

---

[1] Single-Thread
[2] Asynchronous
[3] Thread

فرآیند اجرای یک برنامه در زبان Atalk

۱. در ابتدا در صف همه‌ی اکتورها پیام init وجود دارد.
۲. $i \leftarrow 0$
۳. اکتور iاُم اولین پیام داخل صف (اگر وجود داشته باشد) را بر می‌دارد و کد داخل گیرنده‌ی[4] آن را اجرا می‌کند.
این کد می‌تواند شامل ارسال پیام به بقیه‌ی اکتورها باشد که در این صورت پیام مربوطه به صف اکتور مقصد اضافه می‌شود اما بررسی آن بعداً انجام می‌شود.
۴. اگر در صف اکتورها پیامی نبود اجرای برنامه تمام می‌شود (بهتر است پیاده‌سازی این قسمت را توسط یک counter که در هنگام ارسال پیام یکی اضافه می‌شود و در انتهای بررسی یکی کم می‌شود انجام دهید که صفر شدن آن به معنی عدم وجود پیام در صف اکتورها است).
۵. $i \leftarrow (i + 1) \bmod ActorsCount$
۶. برو به ۳

خطاهای زمان اجرا

- دسترسی غیرمجاز به عناصر آرایه: باید رشته‌ی "IndexOutOfBoundError" (به واسطه‌ی بررسی اتوماتیک، به رشته مشخص شده دقت کنید) در خروجی نوشته و اجرای برنامه متوقف شود.
- ارسال پیام به اکتوری که ظرفیت باقی‌مانده‌ی صف آن تمام شده است: باید رشته "ActorBufforOverflowError" در خروجی چاپ شود امّا اجرای برنامه متوقف نمی‌شود.

بخش امتیازی

همانطور که می‌دانید، یکی از قابلیت‌های زبان Atalk این است که یک اکتور می‌تواند پیامی را به فرستنده پیام درحال پردازش خود ارسال کند (به وسیله کلمه‌ی کلیدی sender). پیاده‌سازی این قابلیت برای این فاز جنبه امتیازی دارد و درصورتی که آن را پیاده‌سازی نکنید، نمره‌ای از شما کسر نخواهد شد (تست‌هایی که برروی کد شما اعمال می‌شود از این قابلیت زبان استفاده نمی‌کنند). در صورتی که این قابلیت را پیاده‌سازی کنید نمره اضافی به شما تعلق خواهد گرفت. توجه داشته باشید درصورتی که قصد پیاده سازی این بخش را دارید، باید خطای زمان اجرای زیر را هم پیاده‌سازی کنید:

- عدم وجود گیرنده‌ی یک پیام در اکتور مقصد هنگام استفاده از کلمه‌ی کلیدی sender[5]: این بررسی لازم است که در اکتور مقصد و هنگام بررسی پیام انجام شود. در صورت خطا باید پیام مربوطه از صف حذف شود و رشته‌ی "InvalidMessageError" در خروجی چاپ شود. همچنین اجرای برنامه متوقف نمی‌شود و ریسمان اجرا دوباره باید در اختیار اکتور کنونی قرار بگیرد

---

[4] Receiver

[5] افرادی که قسمت امتیازی فاز قبل را انجام دادند نباید این قسمت را پیاده‌سازی کنند.

نکات مهم:

- کد های خود را به صورت یک فایل studentID1_studentID2.zip آپلود کنید.
- سوالات خود را در فروم درس مطرح نمایید تا دوستانتان نیز از آنها استفاده کنند.
- دقت کنید که خروجی‌های شما به صورت خودکار تست **می‌شوند**. بنابراین کد اسمبلی نهایی را باید در خروجی استاندارد چاپ کنید.