طراحی کامپایلر

شما به عنوان یه برنامه‌نویس، یه مساله رو جلوی خودتون می‌ذارین، ساختار و الگوریتم‌هاش رو طراحی می‌کنین و بعد هم با نوشتن کدشون، اون‌ها رو پیاده‌سازی می‌کنین. اما آیا کار اینجا تموم می‌شه؟! جواب اینه که نه!

کامپیوترها فقط می‌تونن کدهایی به زبان خودشون (زبان ماشین!) رو بفهمن؛ یک زبان ساده با تعدادی عملیات ابتدایی که یه سخت‌افزار می‌تونه اون‌ها رو مستقیما انجام بده. از طرفی کد ما معمولا به یک زبان «سطح بالا» نوشته می‌شه. ینی زبونی با ساختارهای پیچیده‌تر که به زبان ما انسان‌ها نزدیک‌تره و فهم و نوشتنش هم برامون راحت‌تر.

حالا کی می‌خواد چیزی که ما نوشتیم رو به چیزی که کامپیوتر می‌فهمه ترجمه کنه؟ نرم‌افزاری به نام «کامپایلر»! این درس می‌خواد شما رو با مراحل ساخت یک کامپایلر و ساختار درونی اون آشنا کنه.

هر کامپایلر از لحاظ عملکرد از دو بخش تشکیل شده: بخش تحلیل و بخش تولید.

تو بخش «تحلیل»، برنامه با توجه به گرامر اون زبان خاص، به اجزای سازنده‌ش شکسته می‌شه و اگه توی این تجزیه و تحلیل، از نظر لغوی، نحوی یا معنایی خطایی وجود داشته باشه، به برنامه‌نویس اطلاع می‌ده. تا اینجا خود کامپایلر فهمیده که ما چه چیزی نوشتیم. از اینجا به بعد باید بر اساس چیزی که فهمیده، کد معادلی به زبان ماشین «تولید» کنه که ماشین مقصد هم بتونه منظور ما رو بفهمه و در نهایت اون رو اجرا کنه.

یکی دیگه از اهداف این درس، اینه که درک کنید پیاده‌سازی بعضی از ویژگی‌های ساده‌ی زبان‌ها چه هزینه‌ی محاسباتی‌ای برای کامپیوتر داره و چطور می‌شه این هزینه‌ها رو کاهش داد یا در واقع کامپایلر رو بهینه‌سازی کرد.

این درس توی دانشکده‌ی ما چهار فاز پروژه داره که سه فاز اولش هر کدوم به یکی از بخش‌های تحلیل (لغوی، نحوی یا معنایی) مربوط می‌شن و فاز آخر به تولید کد مقصد.

کامپایلر درس خیلی جذابیه؛ به شرطی که با پروژه‌هاش پیش بیاید و تو تمام مراحل تمیز کد بزنید.