پیش‌گزارش دستورکار پنجم آزمایشگاه ریزپردازنده و زبان اسمبلی

نگار موقتیان، ۹۸۳۱۰۶۲

۱. توضیحات مختصر دربارهٔ دستورات LDR، MOV و STR

- دستور LDR: سینتکس اصلی این دستور به صورت LDR Rd, [Rx] می‌باشد و برای لود کردن محتوای مکانی که Rx به آن اشاره می‌کند درون رجیستر Rd استفاده می‌شود. به علاوه این دستور به عنوان یک شبه دستور به فرم LDR Rd, =32-bit_immediate_value نیز به کار می‌رود که برای لود کردن مقادیر ثابت ۳۲ بیتی درون رجیسترها استفاده می‌شود (به دلیل محدودیت دستور MOV که تنها برای ثابت‌های 8 بیتی قابل استفاده است).
- دستور MOV: سینتکس اصلی این دستور به صورت MOV Rn, Op2 می‌باشد که در آن Op2 می‌تواند یک مقدار ثابت و یا یک رجیستر باشد. این دستور برای کپی کردن مقدار Op2 درون رجیستر Rn به کار می‌رود.
- دستور STR: سینتکس اصلی این دستور به صورت STR Rd, [Rx] می‌باشد و برای ذخیره‌سازی محتوای رجیستر Rd در مکانی که Rx به آن اشاره می‌کند استفاده می‌شود (چیزی برعکس دستور LDR).

* دستورات بالا در حالت پایه بیان شده‌اند و همگی حالت شرطی و ... نیز دارند.

۲. ایده‌ای برای پیاده‌سازی تابع تاخیر در زبان اسمبلی

برای انجام این کار می‌توانیم از یک حلقه استفاده کنیم. می‌دانیم هر دستور (از جمله دستورات مقایسه و ADD) برای اجرا در پردازنده نیاز به چندین کلاک دارد، بنابراین اگر یک متغیر با مقدار اولیهٔ صفر در نظر بگیریم و در یک حلقه هر بار مقدار آن را تا یک مقدار مشخص افزایش دهیم می‌توانیم تاخیری با مقداری تقریباً ثابت ایجاد کنیم. هر چه عدد ماکسیممی که برای این متغیر در نظر می‌گیریم بیش‌تر باشد، مقدار تاخیر ایجاد شده نیز بیش‌تر خواهد بود.

کد اسمبلی مربوط به این تابع مانند زیر می‌باشد (در این جا مقدار ماکسیمم برابر با 0x0040000 در نظر گرفته شده و رجیستر R4 به عنوان متغیر شمارنده استفاده شده‌است):

```
delay
     MOV R4, #0
     LDR R5, =0x0040000
delay_loop
     ADD R4, R4, #1
     CMP R4, R5
     BNE delay_loop
     BX LR
```

۳. پاسخ به پرسش‌های بخش مقدمه

دستورات موجود در Keil برای Build کردن برنامه:

- بخش stop build: برای متوقف کردن روند build در حال اجرا به کار می‌رود.
- بخش batch build: دستور build را بر روی target های پروژهٔ انتخاب شده اجرا کرده و این target ها را لینک می‌کند.
- بخش rebuild: تمام فایل‌های source برنامه را دوباره ترجمه کرده و برنامه را build می‌کند.
- بخش build: تمام فایل‌هایی که تغییر کرده‌اند را ترجمه کرده و برنامه را build می‌کند.
- بخش translate: فایلی که در حال حاضر فعال است را ترجمه می‌کند.

توضیح بخش reset-handler از فایل startup:

به طور کلی توابع نوشته شده در این بخش با پسوند handler برای رسیدگی به وقفه‌های موجود در سیستم استفاده می‌شوند. Reset-handler نیز اولین تکه کدی است که پس از ریست شدن سیستم (با آمدن وقفهٔ اول یا همان reset) اجرا شده و یکسری از تنظیمات ابتدایی (مانند تنظیمات مربوط به استک) را انجام می‌دهد. با آمدن این وقفه اجرای دستورات موجود در پردازنده متوقف می‌شود و با برطرف شدن آن اجرای برنامه از آدرسی که vector table آن را مشخص کرده (در اینجا همان reset-handler) دوباره شروع می‌شود.