**پیش­گزارش دستورکار پنجم آزمایشگاه ریزپردازنده و زبان اسمبلی**

نگار موقتیان، 9831062

1. **توضیحات مختصر دربارۀ 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 32 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.

1. **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 اولیۀ 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

1. **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 پروژۀ 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 وقفۀ 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.