دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

3/7/2021

Homework 1

Lec 1-4

MICROPROCESSOR
AND
ASSEMBLY LANGUAGE

Spring 2021

1) تفاوت های Microprocessor و Microcontroller چیست ؟

طبق صفحه ی آخر اسلاید اول میتوان گفت که:

1- Microprocessor را میتوان قلب یک سیستم کامپیوتری به حساب آورد و Microcontroller را قلب یک سیستم نهفته به حساب آورد.

2- Microprocessor به تنهایی فقط یک پراسسور خالی میباشد، حافظه و سایر دستگاه های I/O از خارج از آن بهش باید وصل شوند. ولی Microcontroller شامل پراسسور و حافظه و دستگاه های I/O باهم میباشد و به صورت یک پکیج به صورت داخلی به هم متصل اند.

3- در Microprocessor چون حافظه و I/O ها به صورت خارجی به پراسسور وصل میشوند پس مدار نهایی بزرگ میباشد ولی در Microcontroller چون حافظه و I/O ها به صورت داخلی و یک مدار مجتمع به هم وصل هستند پس مدار نهایی کوچک میباشد.

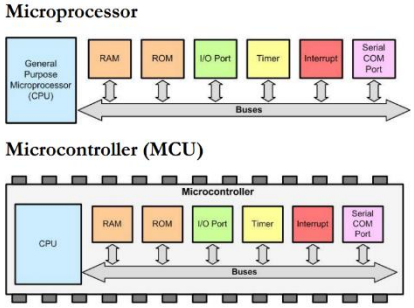
4- از Microprocessor نمیتوان در یک سیستم جمع و جور استفاده کرد چرا که طبق آنچه که گفتیم مدار نهایی بزرگ است ولی Microcontroller برای استفاده در سیستم های کوچک بسیار کارا و مناسب میباشد.

5- مصرف انرژی نهایی در استفاده از Microprocessor با توجه به اجزای خارجی ای که به آن وصل میشوند در کل زیاد است ولی در Microcontroller مصرف انرژی کم و متناسب میباشد از این رو برای سیستم‌هایی که با باتری کار میکنند بسیار مناسب میباشد.

6- دسترسی به حافظه در Microprocessor آهسته است ولی در Microcontroller دسترسی سریع تر میباشد.

7- در استفاده از Microprocessor طراحی معماری مدار براساس معماری Von Neumann میباشد یعنی درواقع instruction و data در یک ماژول حافظه یکسان ذخیره میشوند ولی در Microcontroller معماری براساس مدل Harvard میباشد و instruction و data از هم جدا میباشند.

همچنین تفاوت ساختاری این دو را میتوانید در شکل روبرو مشاهده کنید:



2) حالت‌های مختلف پایه های واحد GPIO را نام ببرید و حالت Pull-up را شرح دهید.

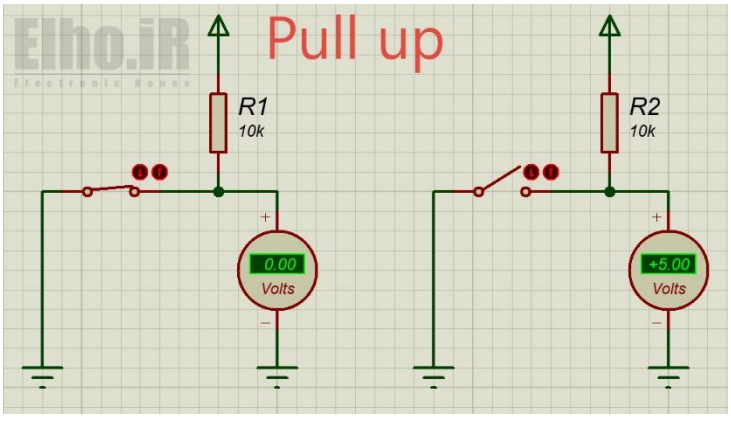
طبق صفحه 7 اسلاید 3 حالت های مختلف پایه های واحد GPIO به صورت زیر میباشند:

1- Pull-up
2- Input Schmitt triggers
3- Multi-drive (open-drain)
4- Glitch filters
5- Debouncing
6- Input change interrupt

حال به شرح حالت Pull-up میپردازیم:

در مدار Pull-up مدار از یک سمت (ورودی) به ولتاژ 0 یا دراقع GND متصل میباشد و طرف دیگر در خروجی را به VCC وصل میکنند پس در واقع میتوان گفت معمولا پایه های Output را Pull-up میکنند.

وقتی Pull-up میکنیم اتفاقی که میوفتد این است که وقتی هیچ دیتایی نداریم و درواقع زمانی که کلید وصل نیست در خروجی سیگنال 1 منطقی (یا مقدار 5 ولت) داریم . زمانی که کلید وصل شود خروجی 0 منطقی (یا مقدار 0 ولت) میشود. در شکل زیر میتوانید این مفهوم را مشاهده کنید:



3) اگر سطح دسترسی ما کاربر باشد اما دستوری را به پردازنده بدهیم که نیاز به سطح دسترسی privilege داشته باشد، چه اتفاقی رخ می‌دهد؟

اگر سطح دسترسی ما کاربر باشد نمیتواند دستورات privileged را در آن سطح اجرا کند و exceptionمیدهد. در واقع همچین دستوری در سطح کاربر غیرمجاز میباشد و اجرا نخواهد شد. برای انجام این دستورات نیاز است که در مود Handler یا همان مود Privileged باشیم.

اگر واقع نیاز داریم که این کد نوشته شده را که به سطح privileged نیاز دارد را اجرا کنیم؛ باید این دستور به عنوان ISR نوشته شود و وقتی در مود کاربر هستیم نمیتوان به راحتی مود را به حالت Privileged تغییر داد و دستور مورد نظر را انجام داد. در ابتدا برفرض ما در مود کاربر هستیم و برنامه ی عادی ما در همان مود اجرا میشود تا اینکه یک اینتراپت یا exception بیاید و سپس state پراسسور به صورت اتوماتیک در stack ذخیره میشود و به مود Handler برویم. سپس اگر دستور را به عنوان ISR آن اینتراپت نوشته باشیم و همچنین اولویت اینتراپت نسبت به کار قبلی بیشتر باشد و Mask هم نشود، آنگاه آن دستور ما که نیاز به این سطح از دسترسی داشت اجرا میشود. سپس پس از انجام آن به مود قبلی برمیگردیم و state ای که ذخیره کرده بودیم بازیابی میشود .

4) به پرسش های زیر پاسخ دهید:

الف) Interrupt Masking چیست و فواید آن را شرح دهید.

ب) فرض کنید می‌خواهیم وقفه شماره 1 را Mask کنیم و وقفه شماره 12 را فعال کنیم و اولویت آن را به 5 تغییر دهیم. محتوای کدام رجیسترهای NVIC تغییر می‌کند؟

الف) پردازنده ها معمولاً دارای یک interrupt mask register داخلی هستند که امکان فعال سازی و غیرفعال کردن وقفه های سخت افزاری به ما میدهد. هر سیگنال وقفه با یک بیت در mask register مرتبط است. در برخی از سیستم ها ، وقفه زمانی که این بیت set میشود، فعال می شود و هنگامی که بیت پاک شود غیرفعال می شود ، در حالی که در برخی دیگر ،اگر این بیت set شود وقفه را غیرفعال می کند. وقتی وقفه غیرفعال شود ، پردازنده سیگنال وقفه مرتبط را نادیده می گیرد. به سیگنالهایی که تحت تأثیر mask قرار می گیرند maskable interrupts گفته می شود. برخی از سیگنال های وقفه تحت تأثیر interrupt mask قرار نمیگیرند و بنابراین نمی توانند غیرفعال شوند. این وقفه ها را می توان non-maskable interrupts (NMI) نامید. NMI ها eventهای با اولویت بالا را نشان می دهند که تحت هیچ شرایطی قابل چشم پوشی نیستند ، مانند سیگنال timeout از watchdog timer. همچنین Mask کردن یک وقفه به معنای غیرفعال کردن آن است ، در حالی که unmask کردن وقفه ، به معنی فعال کردن آن است.

(به طور خلاصه میتوان گفت یک ویژگی پردازنده میباشد که به آن اجازه می دهد تا درخواست interrupt را تا زمانی که بیت mask غیرفعال شود ؛ نادیده بگیرد (mask می کند).)

از فواید آن میتوان گفت برای مثال در قسمت critical section کد هستیم و نمیخواهیم کسی در این فاز برنامه را متوقف کند و میتوان با یک دستور کاری کرد که اگر اینتراپتی هم آمد پردازنده کار ما را متوقف نکند و پس از خارج شدن از این فاز مهم به حالت اول برگردیم و اینتراپت ها فعال شوند.

ب) برای Mask کردن وقفه شماره یک باید محتوای Mask Register را تغییرداد.

برای فعال کردن وقفه شماره 12 باید بیت 12 ام از ISER0 را تغییر دهیم. (میدانیم که 8 تا رجیستر 32 بیتی به اسم Interrupt Set Enable Register یا ISER داریم که هر بیت متعلق به یکی از اینتراپت ها است.)

60 تا رجیستر 4 بایتی برای تعیین اولویت اینتراپت ها داشتیم که بنابراین باید مقدار رجیستر سوم را تغییر دهیم و در بایت چهارم آن اولویت وقفه شماره 12 را باید تغییر دهیم. به این رجیسترها Interrupt Priority Register یا IPR میگویند.

5) 4 حالت وقفه در NVIC را نام برده و شرح دهید. همچنین 5 مورد از سیاست ها و قابلیت‌های آن برای سازماندهی وقفه ها را نام ببرید.

4 حالت وقفه در NVIC به صورت زیر میباشند:

1- **Inactive** : زمانی که برای دیوایس وقفه و اینتراپت تعریف شده ولی الان خبری از آن وقفه نیست. یعنی وقفه نه به صورت active و نه به صورت pending میباشد.

2- **Pending** : در این حالت وقفه منتظر است تا پردازنده به آن سرویس دهد. درواقع با ارسال یک interrupt request از سمت یک peripheral یا یک نرم افزار میتواند حالت متناظر با آن وقفه را به حالت pending ببرد و بنابراین دیوایس اینتراپت را ارسال کرده ولی هنوز پاسخی از سوی پردازنده دریافت نکرده و هنوز وارد ISR نشده است.

3- **Active** : در این حالت پراسسور درحال سرویس دادن به وقفه است و درواقع ISR آن درحال اجرا میباشد ولی هنوز کار آن تمام نشده و درحال اجرا است. دقت شود که همیشه فقط یک وقفه میتواند در حالت active باشد.

4- **Active and Pending(A&P)**: پراسسور درحال سرویس دادن به وقفه یک Divce میباشد و ISR آن درحال اجرا میباشد و همان منبع یا Divice یک وقفه جدید میفرستد که این وقفه به حالت Pending میرود.

حال 5 مورد از قابلیت ها و سیاست های آن برای سازماندهی وقفه ها را در ادامه می آوریم:

1- **Tail-chaining**: زمانی که یک وقفه در حالت active است و یک وقفه‌ی دیگر می‌آید و درحالت pending میرود، زمانی که نوبتش شد دیگر نیاز نیست مقادیر ذخیره شده در رجیسترها را برگردانیم و دوباره آن ها را سیو کنیم که کار بیهوده ایست. کافی است که پشت هم اینتراپت ها را انجام دهیم.

2- **Late-arriving** : اگر بر فرض به خاطر یک وقفه با اولویت پایین کار عادی پردازنده متوقف شد و به خاطر آن به سراغ سیو کردن state فعلی در رجیسترها رفتیم که مثلا 16 کلاک این فرآیند طول میکشد، ولی سپس در همین بین یک اینتراپت با اولویت بالاتر آمد و هنوز اینستراکشن اول اینتراپت قبلی fetch نشده است؛ باید ISR مربوط به اینتراپت جدید را اجرا کنیم با اینکه دیرتر آمده است.

3- **Dynamic reprioritization of interrupts**: اینتراپت هایی را که در NVIC تعریف و شماره گذاری کردیم را به صورت پویا و کاملا نرم افزاری و در حین اجرای برنامه اولویتشان را جا به جا کنیم.

4- **Configurable number of interrupts**: تعداد وقفه هایی که تعریف میکینم میتواند بین 1 تا 240 تا باشد و باتوجه به نیازمان به تعدادی که میخواهیم میتوانیم تعریف کنیم. پس تعداد اینتراپت ها قابل تنظیم است.

5- **Priority masking**: میتوان به صورت نرم افزاری و با یک دستور خاص بعض از اینتراپت ها را Mask کرد تا در حین اجرای یک کد خاص یا critical section برنامه را متوقف نکنند و مزاحم نشوند.

6- **Configurable number of interrupt priorities**: تعداد اولویت ها نیز قابل تنظیم میباشد و بین 3 تا 8 بیت میتوان برای تعیین آن ها میتوان اختصاص داد.

6) به پرسش های زیر پاسخ دهید:

أ. چه وقفه هایی در معماری Cortex-m3 دارای اولویت ایستا می باشند؟ کاربرد هر کدام از این وقفه ها و جایگاه آن ها در جدول بردار وقفه را بنویسید.

ب. اولین ورودی جدول بردار وقفه برای چیست؟

ج. آیا همه وقفه های پویا، وقفه های خارجی (External Interrupt) می باشند؟

د. آیا می توان اولویت وقفه های پویا را به گونه‌ای پیکربندی کرد که از وقفه های ایستا اولویت بالاتری داشته باشند؟

أ) 3 وقفه ی زیر دارای اولویت ایستا هستند:

1- **Reset:** وقفه‌ی شماره‌ی یک میباشد اولویت آن هم 3- میباشد که بالاترین مقدار اولویت است.

جایگاه: جایگاه آن در جدول بردار وقفه سطر شماره1 میباشد

کاربرد: این اینتراپت asynchronous میباشد و وقتی این اینتراپت بیاید در هر حالتی باشیم پردازنده reset میشود. درواقع پس از آمدن آن ، درهنگام اجرای ISR آن به کمک reset vector پردازنده آدرس اولین دستوری را که باید از آن شروع کند را میابد و اجرا از آنجا دوباره آغاز میشود . این دستورالعمل معمولاً به کد مقداردهی اولیه سیستم منشعب می شود. برای این میباشد که درمواقع حساس که نمیخواهیم سیستم به روند خود ادامه دهد یا اختلالی در عملکرد عادی سیستم رخ میدهد آن را ریست کنیم و مجددا از ابتدا شروع به کار کند.

2-**Non-maskable interrupt:** وقفه‌ی شماره‌ی 2 میباشد. اولویت آن 2- میباشد.

جایگاه: جایگاه آن در جدول بردار وقفه سطر شماره 2 میباشد.

کاربرد: این اینتراپت asynchronous میباشد. فارغ از اینکه چه کاری انجام میدهد تحت هیچ شرایطی نمیتوان آن را نادیده گرفت و متوقف کرد مگر اینکه وقفه ی ریست که اولویت بالاتری دارد رخ دهد.ISR ای که ما برای آن مینویسیم را با این شرایط اجرا میکند.

3-**Hard Fault:** وقفه‌ی شماره‌ی 3 میباشد و اولویت آن 1- میباشد.

جایگاه: جایگاه آن در جدول بردار وقفه سطر شماره 3 میباشد.

کاربرد: این اینتراپت synchronous میباشد یک وقفه است که به دلیل خطا در هنگام normal processing یا exception processing رخ میدهد. پس اگر در هنگام اجرا یک خطا رخ بده مثلا اینستراکشنی بخواهد اجرا شود که اصلا برای سیستم تعریف نشده است؛ این وقفه صدا زده میشود.

ب) طبق صفحه‌ی 23 لکچر4 و صفحه‌ی 8 لکچر 5 میتوان گفت:

در ابتدای جدول بردار وقفه مقدار اولیه‌ی stack و همچنین exception vector های پردازنده تعریف میشوند. آدرس شروع استک و initialization آن در این قسمت است. به صورت پیش فرض استک از خونه‌ی آخر حافظه شروع میشود و به صورت کاهشی به سمت خونه های با آدرس کمتر حافظه می آید. میتوان با دستکاری این فیلد از جدول آدرس شروع استک را عوض کرد. پس به هنگام ریست بالای stack از اولین ورودی جدول بردار وقفه بارگیری میشود.

ج)خیر- برای مثال Usage Fault یا SVCall که به صورت synchronous میباشند؛ وقفه هایی داخلی هستند.

د) خیر نمیتوان- وقفه های ایستا بالاترین اولویت را دارند و این اولویت بالای آن ها هم بی علت نیست. برای مثال ما در هر زمانی باید توانایی ریست کردن را داشته باشیم و اگر یک وقفه اولویت بالاتری از آن داشته باشد، در حین اجرای آن دیگر نمیتوان ریست کرد و اینتراپت ریست نادیده گرفته میشود و این دربعضی موارد میتواند خطرناک باشد.

7) فرض کنید سه وقفه A، B و C با ویژگی های زیر در یک سیستم موجود می باشد. چنانچه وقفه B اعلان شود و پیش از پایان بارگذاری محتوای رجیستر ها بر روی پشته، وقفه A و C نیز رخ دهند، آنگاه با فرض اینکه برای هر کدام از عملیات های save و load رجیسترها به 15 کلاک نیاز باشد و مقدار PRIGROUP در رجیستر Application Interrupt and Reset Control Register برابر 0b000 باشد به سوال های زیر پاسخ دهید:

أ. ترتیب اجرای وقفه ها را بدست آورید.

ب. بازده CPU را از زمان رسیدن وقفه B تا زمان بازگشت به اجرای اولیه برای دو حالت زیر را حساب کنید:

I. NVIC دارای ویژگی های Late-arriving و Tail-chaining در مدیریت وقفه ها باشد.

II. دارای این دو ویژگی نباشد.

وقفه ها	بیت های متناظر در *Interrupt Priority Register*	*کلاک مورد نیاز*
A	*0b00000010*	***45***
B	*0b00000001*	***130***
C	*0b00000011*	***70***

جواب در صفحه‌ی بعد

أ) در این حالت چون ما هنوز وارد ISR مربوط به اینتراپت B نشدیم و PRIGROUP مربوط به لین اینتراپت ها یکی میباشد، طبق Late Arriving باید ببینیم کدام یک اولویت بیشتری دارد و آن را اول انجام دهیم.

به این ترتیب طبق جدول میتوان گفت اول اینتراپت B و سپس اینتراپت A و بعد اینتراپت C اجرا میشود.

ب) حالت اول)

در این حالت وقتی وقفه‌ی B در ابتدای کار می‌آید باید محتوای رجیسترها save شود که برای آن 15 کلاک میخواهیم. بعد ISR مربوط به B باید اجرا شود که 130 کلاک میخواهد. حال چون Tail chaining داریم دیگر برای اینتراپت های بعدی لازم نیست مقادیر پشته لود و سپس سیو شوند و مستقیم میرویم سراغ اینتراپت بعدی که 45 کلاک برای اجرای آن و بعد C که برای اجرای آن 70 کلاک میخواهد. درآخر باید مقادیر روی پشته را با 15 کلاک load کنیم تا به ادامه اجرای برنامه اولیه بپردازیم:

تعداد کل کلاک‌ها : $15 + 130 + 45 + 70 + 15 = 275$

حال اگر بازده مقدار کار مفید به کل کار باشد و کار مفید ما درواقع کلاک هایی باشد که برای انجام ISR مربوط به وقفه باشد آنگاه:

تعداد کلاک‌های مفید : $130 + 45 + 70 = 245$

$$\text{بازده} = \frac{245}{275} = 0.89 \rightarrow 89\%$$

حالت دوم)

در این حالت چون Late Arrivingنداریم به همان ترتیبی که اینتراپت ها می آیند اجرا میشوند. چون Tail Chaining نداریم باید بین هردو اینتراپت یک بار محتویات روی پشته را load کنیم و سپس save کنیم و بعد بریم سراغ اینتراپت بعد پس در واقع چون 3 اینتراپت داریم دوتا 30 کلاک اضافه میشود:

$$15 + 130 + 15 + 15 + 45 + 15 + 15 + 70 + 15 = 335$$

حال دوباره مقدار کار مفیدمون مانند قبل میباشد. بنابراین داریم:

تعداد کلاک‌های مفید : $130 + 45 + 70 = 245$

$$\text{بازده} = \frac{245}{335} = 0.731 \rightarrow 73.1\%$$

- مهلت ارسال تمرین ساعت 23.55 روز 4 فروردین می‌باشد.
- برای پاسخ به پرسشهای این تمرین میتوانید در صورت نیاز به فصل 5 و 8 مرجع فنی Cortex-m3 که در مودل بارگزاری شده است مراجعه کنید.
- سوالات خود را می‌توانید تنها از طریق ایمیل AUTMicroTA@gmail.com بپرسید.
- ارائه پاسخ تمرین بهتر است به روش های زیر باشد:
  1) استفاده از فایل docx. تایپ پاسخ‌ها و ارائه فایل Pdf
  2) چاپ تمرین و پاسخ دهی به صورت دستنویس خوانا
- فایل پاسخ تمرین را تنها با قالب **HW1- 9731***.pdf** در مودل بارگزاری کنید.
  - نمونه: HW1- 9731747
- فایل زیپ ارسال **نکنید**.