



دانشکده مهندسی  
کامپیوتر و فناوری اطلاعات

10/12/2021



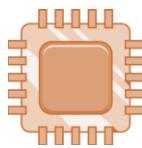
## Homework 1

Lec 1-5



MICROPROCESSOR  
AND  
ASSEMBLY LANGUAGE

Fall 2021



# MICROPROCESSOR AND ASSEMBLY LANGUAGE

Dr. Farbeh

## Homework 1

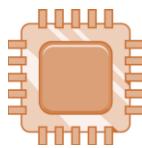


1) به پرسش های زیر در مورد ISA پاسخ دهید:

الف) ISA پردازنده ما باید شامل کدام گروهها از فانکشن ها باشد تا ISA کاملی به حساب آید؟

ب) ISA چه ویژگی هایی از پردازنده را مشخص می کند (حداقل به سه مورد اشاره کنید)؟

مثال: Cisc یا Risc بودن پردازنده

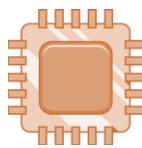


(2) به سوالات ریز در رابطه با Microcontrollers پاسخ دهید:

الف) میکرو ای که ما در درس استفاده می کنیم (SAM3X8E) از کدام یک از معماری های Harvard یا Von Neumann استفاده می کند و دلایل آن چیست (دو دلیل)؟

ب) چند تا از برتری هایی که باعث شده ست در سیستم های نهفته از Microcontroller استفاده شود را نام ببرید (سه مورد کافی است).

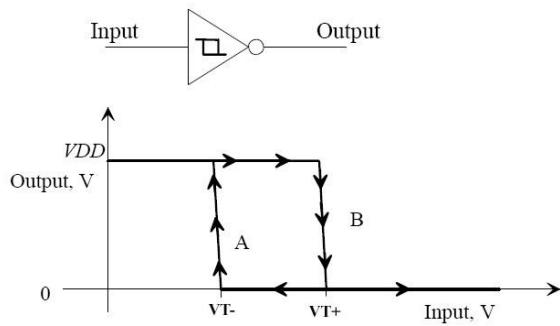
ج) حالت های مختلف میکرو (SAM3X8E) در Low Power Modes را نام ببرید و برای هر کدام یکی از مواقع استفاده را مثال بزنید.

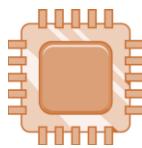


(3) به سوالات زیر در مورد اجزای ریزپردازنده (SAM3X8E) پاسخ دهید:

الف) سه مدل مختلف تایмер در این ریزپردازنده را نام ببرید و موارد استفاده از هر کدام را شرح دهید.

ب) شکل زیر نشان دهنده کدام GPIO میکرو ماست و نمودار آن را توضیح دهید.

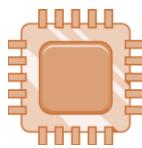




4) به پرسش های زیر در مورد وقفه های تودرتو پاسخ دهید:

الف) NVIC چگونه وقفه های تودرتو را مدیریت می کند (از دیدگاه رجیستر های NVIC شرح دهید)?

ب) چهارتا از دستوراتی که تعداد کلک بالایی برای اجرا نیاز دارند را نام ببرید و اگر در حین پردازش این دستورات وقفه ای رخ دهد چگونه با آنها برخورد خواهد شد؟



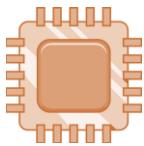
5) به سوالات زیر در مورد NVIC پاسخ دهید:

الف) دلیل وجود دو حالت مختلف Active و A&P برای وقفه‌ها در NVIC را شرح دهید.

ب) فرق بین دو ویژگی Tail-chaining و Late-arriving را توضیح دهید.

ج) دلایل وجود قابلیت Masking را نام ببرید و انواع حالاتی که می‌توانیم با استفاده از رजیسترهای CPU

جمعی از وقفه‌ها را باهم Mask کنیم را شرح دهید.



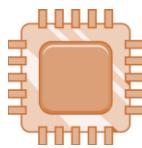
6) به پرسش‌های زیر در مورد NVIC Register پاسخ دهید:

الف) Vector table چیست و محتوی آن چگونه است؟

ب) چرا وقفه‌های NVIC در سری ریزپردازنده‌های ARM از شماره 1 شروع می‌شوند نه 0؟

ج) با توجه به اینکه رجیستر NVIC-IPR اعداد Unsigned را در خود ذخیره می‌کند چگونه وقفه‌هایی با

اولویت منفی داریم؟



- مهلت ارسال تمرین ساعت 23.59 روز 30 مهر 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 از طریق تلگرام از تدریسیارهای گروه خود بپرسید.
- ارائه پاسخ تمرین بهتر است به روش‌های 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\_StudentNumber\_G[groupnumber].pdf** در مودل بارگزاری کنید.
- نمونه: Hw1\_9731121\_G1
- فایل زیپ ارسال نکنید.



دانشکده مهندسی  
کامپیوتر و فناوری اطلاعات

10/12/2021



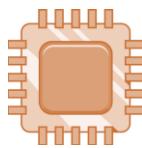
## Homework 1

Lec 1-5



MICROPROCESSOR  
AND  
ASSEMBLY LANGUAGE

Fall 2021



(1) به پرسش های زیر در مورد ISA پاسخ دهید:

الف) ISA پردازنده ما باید شامل کدام گروهها از فانکشنها باشد تا ISA کاملی به حساب آید؟

پاسخ:

برای اینکه یک ISA کامل باشد باید دارای حداقل این سه گروه از فانکشنها باشد:

- Load / Store •
- Control •
- Arithmetic / Logic •

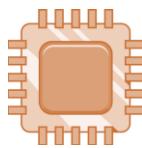
با این سری از دستورات ما می‌توانیم تمامی دستورات مورد استفاده در زبان‌های سطح بالاتر را تولید کنیم.

ب) ISA چه ویژگی‌هایی از پردازنده را مشخص می‌کند (حداقل به سه مورد اشاره کنید)؟

مثال: Cisc یا Risc بودن پردازنده

پاسخ:

1. طول دستورات (دستورات چند بیتی خواهند بود)
2. طول دستورات ثابت یا متغیر (براساس Cisc یا Risc بودن پردازنده)
3. تعداد رجیسترها و تعداد بیت‌های آنها
4. محل قرارگیری Operand ها (رجیستر یا پشته یا حافظه)
5. نحوه برقراری ارتباط با حافظه (ممکن است چند چند مدل Load از حافظه داشته باشیم)
6. فرمت دستورات



(2) به سوالات ریز در رابطه با Microcontrollers پاسخ دهید:  
الف) میکرو ای که ما در درس استفاده می کنیم (SAM3X8E) از کدام یک از معماری های Harvard یا Von Neumann استفاده می کند و دلایل آن چیست (دو دلیل)؟

پاسخ:

میکرو ما از معماری Harvard پیروی می کند.

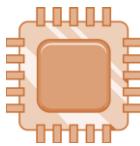
دلیل 1:

با توجه به اینکه میکرو ما بیشتر برای سیستم های نهفته استفاده می شود و مانند کامپیوترهای روزمره نیازی ندارد که هی برنامه ای که روی رم آن قرار دارد تغییر کند و معمولاً برنامه ای که روی آنها قرار میگیرد تا مدت زمان زیادی نیاز به تغییر ندارد درنتیجه که این دو حافظه از هم جدا باشند سرعت پردازش ما بیشتر می شود.

دلیل 2:

با توجه به صحبت های مطرح شده در کلاس می دانیم که پردازنده های کامپیوتر های رومیزی ما زیر به طور مخفی و در cache از معماری Harvard استفاده می کنند و با توجه به اینکه میکرو ما اصلاً cache ندارد و از حافظه کوچکی برخوردارست و دلیلی که معماری Von Neumann جوابگو خواهد وجود cache است پس در این پردازنده ها بهتر از معماری Harvard استفاده کنیم. با توجه به نبود قابلیت اجرا پایپ لاین وجود ندارد و برای اینکه این قابلیت وجود داشته باشد باید از معماری cache استفاده کنیم. دلیل نبود cache هم آن است که حافظه اصلی خودش به اندازه کافی بزرگ نیست که نیاز به cache و طبق دلیل 1 برنامه برروی حافظه دستورات زیاد تغییر نمی کند پس وجود cache منطقی نیست.

**نکته:** با توجه به اینکه دلیل 1 بیشتر در بخش اسambilی درس مطرح می شود نوشتن آن ضرورتی ندارد و نوشتن دلیل 2 کافی است.



ب) چند تا از برتری هایی که باعث شده ست در سیستم های نهفته از **Microcontroller** استفاده شود را نام ببرید (سه مورد کافی است).

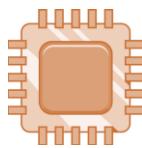
پاسخ:

- تمامی حافظه ها و I/O ها در درون یک میکروکنترلر قرار دارد و نیازی نیست این دیوایس ها را خریداری کرده و با هم اسمبل کنیم، چون بیشتر هدف ما از خرید میکروکنترلر یک واحد محاسباتی خالی نیست و بیشتر هدف استفاده خاص منظوره از آنها در یک سیستم بزرگتر است.
- با توجه به اینکه همه دیوایس ها را دارد مدار کوچکی دارد و مناسب فضای با اندازه کوچک است
- دسترسی به حافظه سریعی دارند
- قدرت پردازشی زیادی ندارند و به همین دلیل ارزان هستند و مناسب برای سیستم های نهفته
- صرف کمتر انرژی

ج) حالت های مختلف میکرو (SAM3X8E) در Low Power Modes را نام ببرید و برای هر کدام یکی از موقع استفاده را مثال بزنید.

پاسخ:

- peripherals: Backup Mode
- هاست. مثلا وقتی که ماشین لباسشویی خاموش است و ما آن را روشن می کنیم .
- : Wait Mode
- های Real-Time می تواند استفاده شود .
- : Sleep Mode



(3) به سوالات زیر در مورد اجزای ریزپردازنده (SAM3X8E) پاسخ دهید:

الف) سه مدل مختلف تایمر در این ریزپردازنده را نام ببرید و موارد استفاده از هر کدام را شرح دهید.

پاسخ:

• یک شماره 32 بیتی است، ثانیه های سپری شده را می شمرد به همین دلیل به

آن Real-time می گویند. می تواند برای وقفه های دوره زمانی ثابت استفاده شود. (مثلا 10 ثانیه ای)

• این تایمر برای میکرو ساعت می سازد. یعنی هم ثانیه دارد هم دقیقه هم ساعت

و هم تقویم و می تواند به ما دقیق این موارد را بگویید. این تایмер یه زمان اولیه دارد و بعد از شروع به

شماردن می کند و هر وقت برق مدار قطع شود ریست می شود. فرق آن با RTT این است که،

RTC صرفا ثانیه ها را می شمارد و برای محاسبه زمان باید خودمان برحسب ثانیه ها بدست بیاوریم

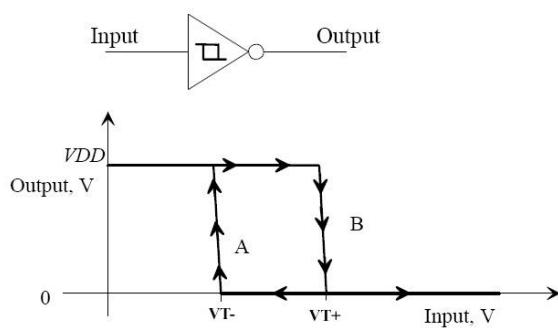
ولی RTC خودش این کار را انجام می دهد.

• تایmer مراقب، سیستم را از Deadlock خارج می کند اگر برنامه ای که در

حال اجرا باشد از تایمر Watchdog بیشتر شود، سیستم را ریست می کند تا از این وضع خلاص

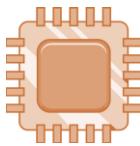
شود.

ب) شکل زیر نشان دهنده کدام GPIO میکرو ماست و نمودار آن را توضیح دهید.



پاسخ:

GPIO که در این نمودار نشان داده شده Triggers Schmitt Input است. این نمودار بیانگر اینست که به فرض ولتاژ ورودی مابین 0 تا 5 ولت باشد و باید آن را به دو عدد دیجیتال 0 و 1 تبدیل کند حال در این مود به این صورت است که اگر ولتاژ از برسد به 2.5، یک تعبیر نمی شد بلکه باید یک مقداری از 2.5 بالاتر برود



(میتوانیم این مقدار threshold را در این مثال 0.5 در نظر بگیریم) یعنی اگر ولتاژ به 3 برسد تازه مقدار دیجیتال آن 1 می شود.

4) به پرسش های زیر در مورد وقفه های تودرتو پاسخ دهید:

الف) NVIC چگونه وقفه های تودرتو را مدیریت می کند (از دیدگاه رجیستر های NVIC شرح دهید)?

پاسخ:

فرض می کنیم وقفه اول در حال اجرا باشد در نتیجه در رجیستر IABR\_NVIC بیت مربوط به وقفه اول بر اساس شماره وقفه اش 1 می باشد. حال وقفه دوم رخ می دهد دو حالت پیش می آید:

الف) وقفه دوم اولویت بالاتر داشته باشد حال در رجیستر IABR\_NVIC بیت مربوط به وقفه اول 0 می شود و بیت مربوط به وقفه دوم 1 می شود (رجیستر IABR\_NVIC در هر زمان حداکثر یک بیت 1 دارد) و سپس در رجیستر ISPR\_NVIC بیت مربوط به وقفه اول 1 می شود و منتظر می ماند تا ISR وقفه دوم کامل اجرا شود. سپس IABR\_NVIC وقفه اول 1 و وقفه دوم 0 می شود و در رجیستر ICPR\_NVIC بیت مربوط به وقفه اول 1 می شود تا این وقفه از حالت pending خارج شود.

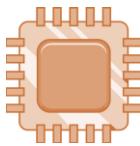
ب) وقفه اول اولویت بالاتر داشته باشد حال در رجیستر IABR\_NVIC بیت مربوط به وقفه اول 1 می باشد و نیازی به تغییر ندارد سپس در رجیستر ISPR\_NVIC بیت مربوط به وقفه دوم 1 می شود و منتظر می ماند تا ISR وقفه اول کامل اجرا شود. سپس IABR\_NVIC وقفه اول 0 و وقفه دوم 1 می شود و در رجیستر ICPR\_NVIC بیت مربوط به وقفه دوم 1 می شود تا این وقفه از حالت pending خارج شود.

ب) چهارتا از دستوراتی که تعداد کلک بالایی برای اجرا نیاز دارند را نام ببرید و اگر در حین پردازش این دستورات وقفه ای رخ دهد چگونه با آنها برخورد خواهد شد؟

پاسخ:

Load Multiple (LDM), Store Multiple (STM), Push, Pop, MULS

در صورتی که وقفه ای در حین اجرا این دستورات رخ دهد و برنامه در شرایط بحرانی نباشد، دستورات کاملا دراپ می شود و بعد از اجرا وقفه دوباره از اول اجرا می شوند.



5) به سوالات زیر در مورد NVIC پاسخ دهید:

الف) دلیل وجود دو حالت مختلف Active و A&P برای وقفه‌ها در NVIC را شرح دهید.

پاسخ:

دلیل وجود حالت A&P این است که اگر یک وقفه در حال اجرا باشد و باز همان دیوایس وقفه جدیدی بفرستد وضعیت آن به A&P تغییر می‌کند.

ب) فرق بین دو ویژگی Tail-chaining و Late-arriving را توضیح دهید.

پاسخ:

Tail-chaining: این ویژگی به این معناست که اگر وقفه‌ای در حال اجرا باشد و وقفه دیگری رخ دهد حال یکی از این وقفه‌ها به حالت pending می‌رود، پس از اتمام یکی از آنها دیگه لازم نیست رجیستر های برنامه اصلی را باز لود کرده و بعد از آن باز سیو کنیم و به سراغ وقفه دوم برویم و پردازندۀ باهش عمل می‌کند و بلافاصله بعد از اتمام وقفه اول به سراغ دومی می‌رود و پس از اتمام آن رجیستر هایی که به صورت سخت افزاری سیو شده بودند را لود می‌کند.

Late-arriving: فرض کنید در حین سیو کردن رجیستر های برای یک وقفه باشیم که یک وقفه با اولویت بیشتر رخ دهد در حین حالت با توجه به اینکه وقفه دوم اولویت بیشتری دارد انجام می‌شود با

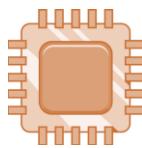
اینکه ما برای وقفه اول رجیستر ها را سیو کردیم (وقفه‌ای که دیر تر رسیده است اجرا می‌شود)

Tail-chaining: مربوط به انتهای اجرا وقفه هاست در حالی که Late-arriving مربوط به ابتدای اجرای آنهاست.

ج) دلایل وجود قابلیت Masking را نام ببرید و انواع حالتی که می‌توانیم با استفاده از رجیسترهای CPU جمعی از وقفه‌ها را باهم Mask کنیم را شرح دهید.

پاسخ:

در واقع Masking یعنی بعضی از وقفه‌ها را نادیده بگیریم و آنها را غیرفعال کنیم و به وقت نیاز باز فعالشان کنیم. مثلا زمانی که می‌خواهیم وارد ناحیه بحرانی یا Critical Section شویم. یک نمونه دیگر نیز وقتی مثلا وقفه‌های مربوط به یک دیوایس خارجی را مدتی قطع می‌کنیم تا روند اصلی برنامه طی شود و از طرف آن دستگاه برای مدتی وقفه نداشته باشیم.



: يك رجيستر تک بيتی که همه وقفه‌های با الويت 0 به بالا را mask می‌کند.

: يك رجيستر 8 بيتی که يك عدد بين 1 تا 240 در آن قرار می‌گيرد و از آن اولويت به بعد mask می‌شود.

: يك رجيستر تک بيتی کاملا مشابه با Primask صرفا با اين تفاوت که وقفه‌هایي با اولويت 1- به بالا را mask می‌کند. (وقفه HardFault که يك وقفه سختافزاری نیز هست)

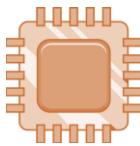
6) به پرسش‌های زیر در مورد NVIC Register پاسخ دهيد:

الف) Vector table چیست و محتوی آن چگونه است؟

پاسخ:

يک جدول 16 ردیفی است که هر ردیف 32 برای شامل می‌شود (DCD) و در آن آدرس شروع ISR مربوط به وقفه مورد نظر با هر خانه را نشان می‌دهد. الان مثلا خانه شماره 1 آدرس وقفه Reset ISR را نشان می‌دهد.

<u>Vectors</u>	<b>DCD</b>	<u>__initial_sp</u>	<b>; Top of Stack initialization</b>
	<b>DCD</b>	<u>Reset_Handler</u>	<b>; Reset Handler</b>
	<b>DCD</b>	<u>NMI_Handler</u>	<b>; NMI Handler</b>
	<b>DCD</b>	<u>HardFault_Handler</u>	<b>; Hard Fault Handler</b>
	<b>DCD</b>	<u>MemManage_Handler</u>	<b>; MPU Fault Handler</b>
	<b>DCD</b>	<u>BusFault_Handler</u>	<b>; Bus Fault Handler</b>
	<b>DCD</b>	<u>UsageFault_Handler</u>	<b>; Usage Fault Handler</b>
	<b>DCD</b>	<u>SecureFault_Handler</u>	<b>; Secure Fault Handler</b>
	<b>DCD</b>	0	<b>; Reserved</b>
	<b>DCD</b>	0	<b>; Reserved</b>
	<b>DCD</b>	0	<b>; Reserved</b>
	<b>DCD</b>	<u>SVC_Handler</u>	<b>; SVCall Handler</b>
	<b>DCD</b>	<u>DebugMon_Handler</u>	<b>; Debug Monitor Handler</b>
	<b>DCD</b>	0	<b>; Reserved</b>
	<b>DCD</b>	<u>PendSV_Handler</u>	<b>; PendSV Handler</b>
	<b>DCD</b>	<u>SysTick_Handler</u>	<b>; SysTick Handler</b>



ب) چرا وقفه‌های NVIC در سری ریزپردازندۀ‌های ARM از شماره 1 شروع می‌شوند نه 0؟

پاسخ:

اولین خانه Vector table با شماره 0 مربوط به آدرس شروع Stack است و وقفه‌ها به ترتیب از خانه شماره 1 به بعد شروع می‌شوند برای همین اولیه وقفه ما شماره 1 دارد نه 0.

به صورت پیش‌فرض Stack از خانه آخر حافظه شروع می‌شود و به صورت کاهشی آدرس آنها کم می‌شود با هر Push و ما با دستکاری خانه شماره 0 این جدول می‌توانیم این پیش‌فرض اولیه را تغییر دهیم.

ج) با توجه به اینکه رجیستر NVIC-IPR اعداد Unsigned NVIC را در خود ذخیره می‌کند چگونه وقفه‌هایی با اولویت منفی داریم؟

پاسخ:

برای این وقفه‌های ایستا مانند Reset و ... مدار آنها به صورت سخت افزاری پیاده سازی شده است و مانند بقیه وقفه رجیستری ندارند که به آن مقدار دهیم.

- مهلت ارسال تمرین ساعت 23.59 روز 30 مهر 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 از طریق تلگرام از تدریسیارهای گروه خود بپرسید.
- ارائه پاسخ تمرین بهتر است به روش‌های 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\_StudentNumber\_G[groupnumber].pdf** در مودل بارگزاری کنید.
- نمونه: Hw1\_9731121\_G1
- فایل زیپ ارسال نکنید.

10/28/2021



---

## Homework 2

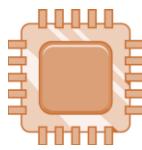
### Lec 6-8

---



MICROPROCESSOR  
AND  
ASSEMBLY LANGUAGE

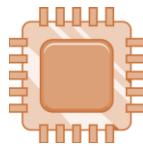
Fall 2021



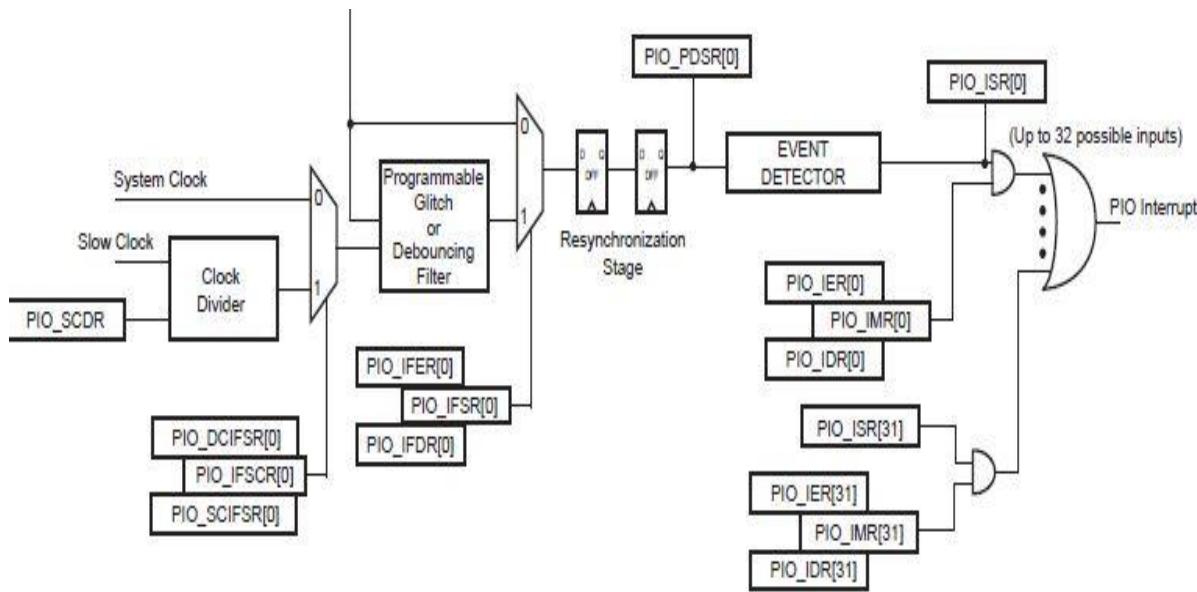
1) الف) مفاهیم زیر را به اختصار توضیح دهید.

- full duplex •
- slave receive •

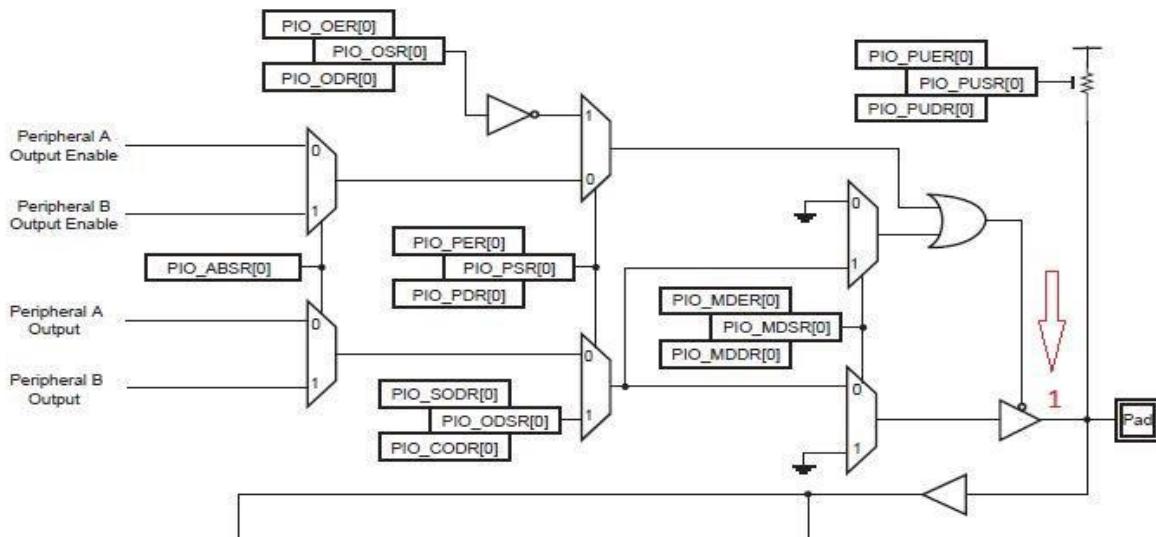
ب) open drain چیست و توضیح دهید کدام پروتکل از این مفهوم تبعیت می‌کند.



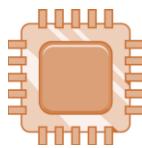
2) الف) در شکل زیر چگونه متوجه شویم کدام پین باعث ایجاد وقفه شده است؟



ب) در شکل زیر اگر  $\text{PIO\_PSR} = 1$  باشد مقادیر خواسته شده را بیابید.



$$\text{PIO\_ODSR} = ?, \text{PIO\_MDSR} = ?, \text{PIO\_OSR} = ?$$



۳) الف) کلاک PIO Controller از کدام بخش تامین میشود و قطع بودن آن چه مزیت و عیبی خواهد داشت؟

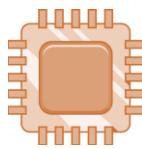
ب) در صورت غیرفعال بودن کلاک واحد PIO Controller، مشخص کنید کدام یک از ویژگی های زیر در PIO Controller فعال و کدام یک غیرفعال است؟

I) نوشتن بر روی رجیستر های PIO Controller

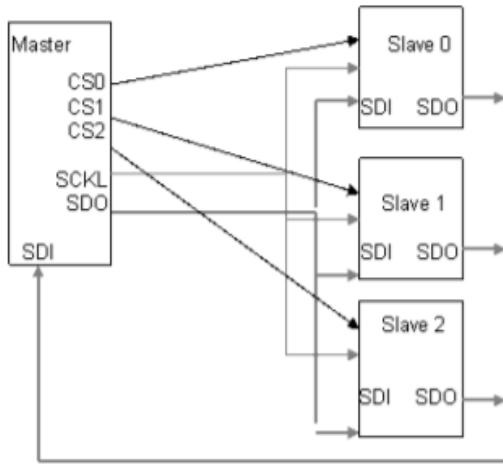
II) خواندن سطح منطقی پین های خروجی

III) استفاده از پین ها برای تولید وقفه

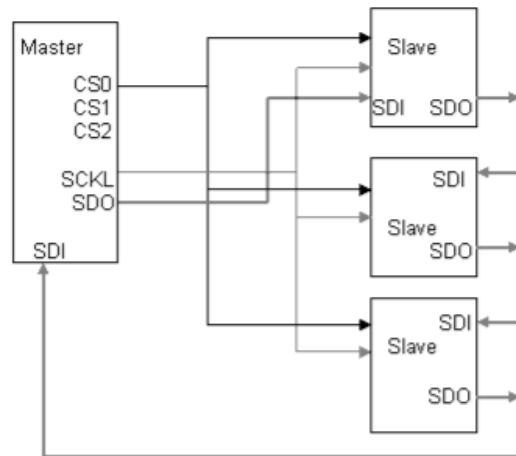
IV) کردن پین ها pull-up



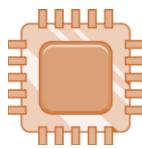
4) برای پیاده سازی یک زنجیره دستگاه ها به کمک رابط SPI، دو تپولوژی زیر پیشنهاد شده است. این دو روش را مقایسه کنید و معایب هر کدام را ذکر کنید.



روش دوم



روش اول



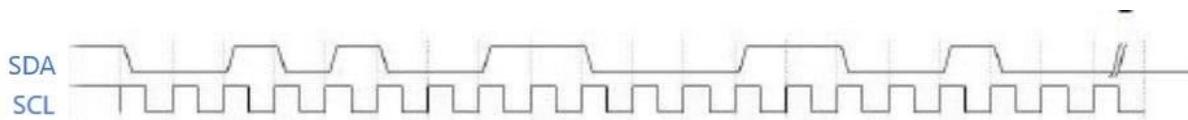
5) الف) سیگنال زیر نمایی از پروتکل I2C با مود ادرس 7بیتی است. مشخص کنید:

I) چه داده ای

II) با کدام ادرس

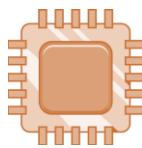
III) به صورت خواندن یا نوشتمن

منتقل میشود.

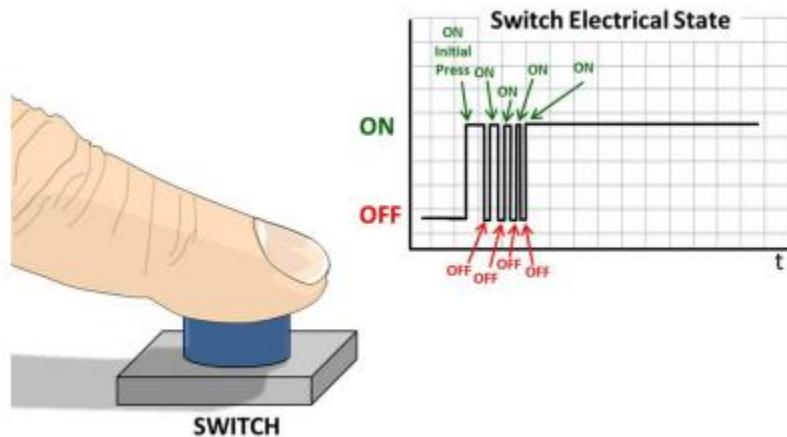


ب) در صورتی که بخواهیم 100 بایت داده را از طریق دو رابط SPI و I2C ارسال کنیم، بازدهی (نسبت تعداد بیت داده به کل بیت ها) را در هر دو رابط محاسبه کنید. بازدهی کدام رابط بیشتر است؟

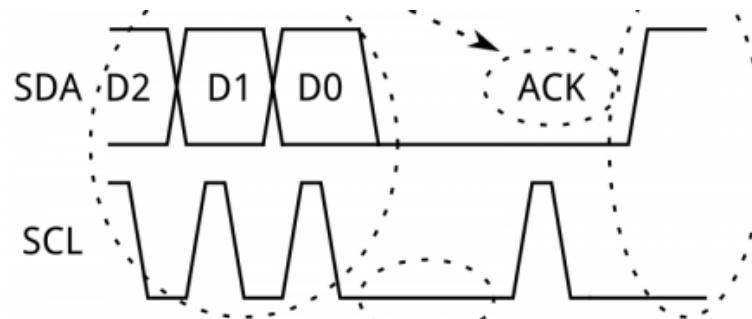
(فرض کنید در رابط SPI در هر frame 16 بیت داده می توان ارسال کرد.)

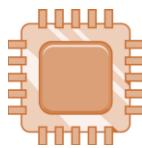


(6) الف) تصویر زیر پدیدهی Switch Bounce را نشان می‌دهد. نمونه‌ای از مشکلات احتمالی ناشی از این نوسان‌ها را بیان کنید و برای دوری از این مشکلات چه راه حلی وجود دارد؟ توضیح دهید.



ب) در مورد مفهوم clock stretching تحقیق کنید.





مهلت ارسال تمرین ساعت 23:55 روز دوشنبه 17 آبان می باشد.

سوالات خود را می توانید از طریق تلگرام از تدریسیارهای گروه خود 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 **HW2 - 9731\*\*\*.pdf** در مدل بارگزاری کنید.

HW2- 9731097 نمونه:

فایل زیپ ارسال نکنید.

10/28/2021



---

## Homework 2

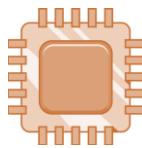
### Lec 6-8

---



MICROPROCESSOR  
AND  
ASSEMBLY LANGUAGE

Fall 2021



۱) الف) مفاهیم زیر را به اختصار توضیح دهید.

- full duplex •
- slave receive •

پاسخ:

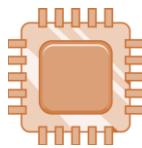
۱- انتقال دیتا ، به صورت دو طرفه ، همزمان روی یک **line** انجام می پذیرد(مانند انتقال دیتا در مکالمات تلفنی)

۲- در حالت C2I مستر ادرس یک **slave** را میفرسد و اعلام میکند داده ای قرار است ارسال شود سپس داده را میفرستد.

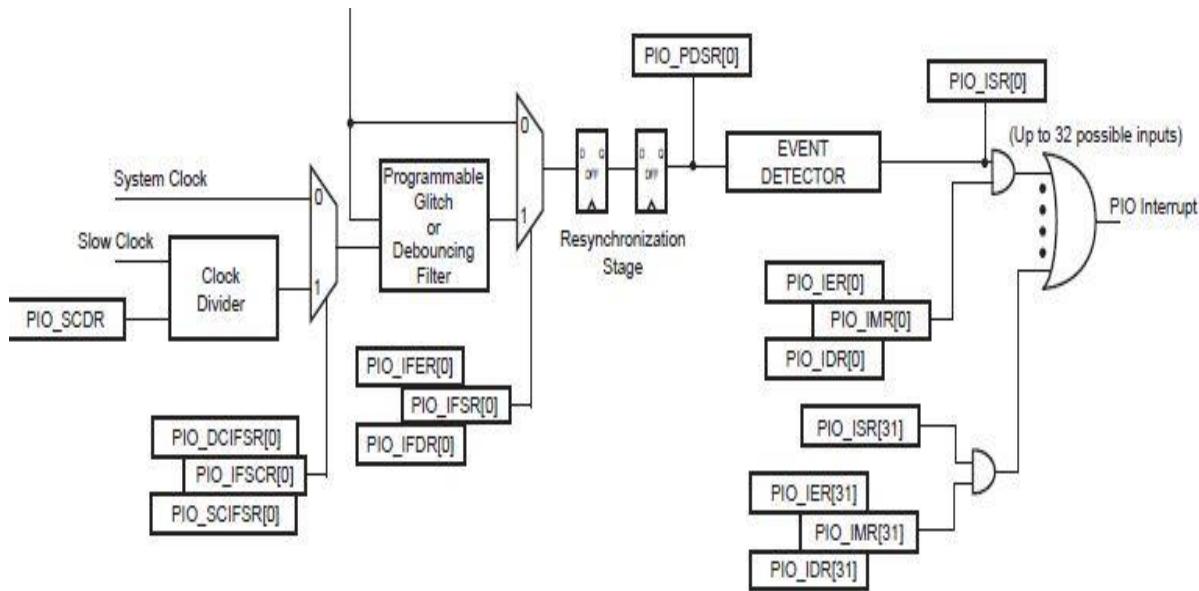
ب) **open drain** چیست و توضیح دهید کدام پروتکل از این مفهوم تبعیت میکند.

پاسخ:

هیچ اختلال در بس نمیتواند به وجود بیاید که در آن یک دستگاه در تلاش باشد تا خط را بالا بکشد(۱) در حالی که دیگری میخواهد آن را پایین بکشد(۰)، و در نتیجه، احتمال آسیب به درایورها یا اتصال توان بیش از حد در سیستم از بین میرود. هر خط سیگنال دارای یک مقاومت پول آپ میباشد، که سیگنال را هنگامی که هیچ دستگاهی آن را پایین نمیکشد بالا نگه دارد. پروتکل C<sup>2</sup>I از این مفهوم تبعیت میکند



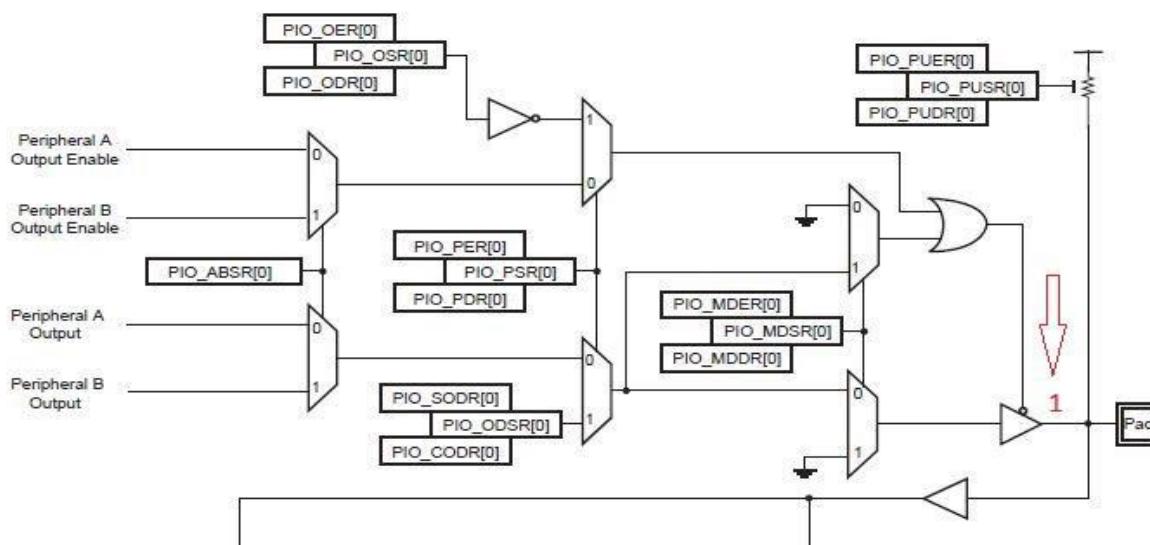
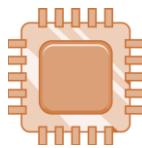
۲) الف) در شکل زیر چگونه متوجه شویم کدام پین باعث ایجاد وقفه شده است؟



پاسخ:

در هر کدام از پین ها یک Event Detector وجود دارد که یک شدن آن به معنای آمدن وقفه می باشد و مقدار آن در بیت متناظر آن پایه رجیستر **PIO\_ISR** ذخیره می شود و میتوان به سادگی با نگاه کردن به رجیستر 32 بیتی **PIO\_ISR** متوجه شد بیت چندم آن 1 شده است و میتوان متوجه شد کدام یک از پین ها باعث ایجاد وقفه شده است

ب) در شکل زیر اگر  $\text{PIO_PSR} = 1$  باشد مقادیر خواسته شده را بیابید.



$\text{PIO\_ODSR} = ?, \text{PIO\_MDSR} = ?, \text{PIO\_OSR} = ?$

پاسخ:

$$\text{PIO\_ODSR} = 1$$

مقدار خروجی مدنظر کاربر را مشخص میکند و از آنجایی که بافر سه حالته یک است یعنی از این رجیستر یک به آنجا منتقل شده

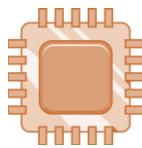
$$\text{PIO\_MDSR} = 0$$

حالت مالتی درایو را مشخص میکند که در اینجا نداریم

$$\text{PIO\_OSR} = 1$$

بافر سه حالته مقدار یک دارد و فعال است یعنی خروجی  $\text{Or}$  برابر با صفر بوده که این در صورتی اتفاق میافتد که  $\text{osr}$  مقدار یک داشته باشد.





(۳) الف) کلاک PIO Controller از کدام بخش تامین میشود و قطع بودن آن چه مزیت و عیبی خواهد داشت؟

پاسخ:

کلاک از واحد controller management power(PMC) تامین میشود

مزیت: قطع بودن کلاک controller PIO باعث ذخیره شدن انرژی میشود(کمتر شدن انرژی مصرفی)

عيوب: با قطع بودن کلاک controller PIO نمیتوانیم از همه ی ویژگی های آن استفاده کنیم برای مثال وقفه ها و حالت های مختلف آن

ب) در صورت غیرفعال بودن کلاک واحد PIO Controller، مشخص کدام یک از ویژگی های زیر در PIO Controller فعال و کدام یک غیرفعال است؟

I) نوشتن بر روی رجیسترها

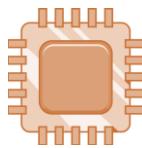
II) خواندن سطح منطقی پین های خروجی

III) استفاده از پین ها برای تولید وقفه

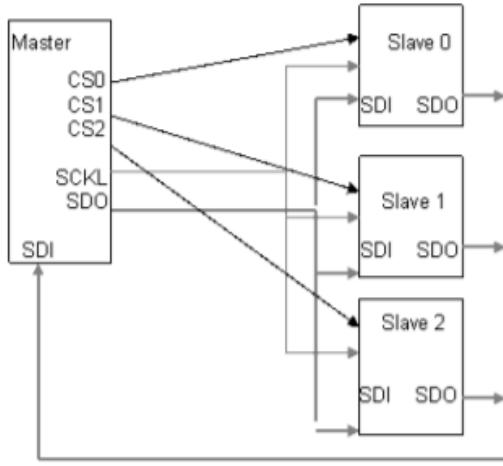
IV) pull-up کردن پین ها

پاسخ:

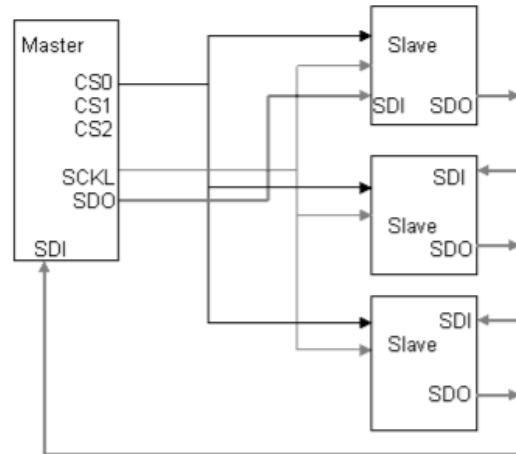
فعال	(I)
غیرفعال	(II)
غیرفعال	(III)
فعال	(IV)



۴) برای پیاده سازی یک زنجیره دستگاه ها به کمک رابط SPI، دو تولوژی زیر پیشنهاد شده است. این دو روش را مقایسه کنید و معاایب هر کدام را ذکر کنید.



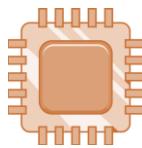
روش دوم



روش اول

پاسخ:

روش اول نیاز به SS (یا cs) کمتری نسبت به روش دوم دارد، بنابراین هزینه پیاده سازی سخت افزاری آن پایینتر است. همچنین جایگذاری قطعات در روش اول ساده تر است چرا که تنها لازم است زنجیره تشکیل شود و هر قطعه به قطعه ای بعدی نزدیک باشد در حالی که در روش دوم، همه ای قطعات باید همچوواری مکانی داشته باشند. در مقابل بازدهی و سرعت روش دوم با لاتر است چرا که در روش اول، داده ها برای رسیدن از هر Slave به Master انتهای مسیر را بروند و از Slave های دیگر عبور کنند. همچنین پیچیدگی در روش اول بیشتر است چرا که برای دسترسی به داده خاص، باید تشخیص داد که داده در کدام دستگاه است و چند کلاک خواندن آن داده نیاز است.

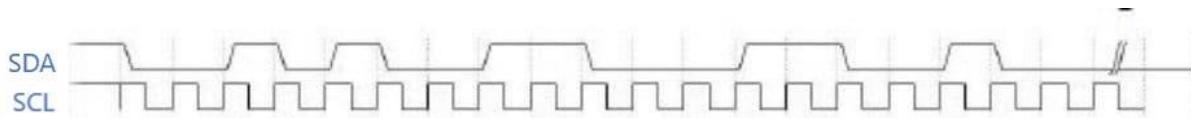


۵) الف) سیگنال زیر نمایی از پروتکل I2C با مود ادرس ۷بیتی است. مشخص کنید:

I) چه داده ای: 00110010

II) با کدام ادرس: 0101001

III) به صورت خواندن یا نوشت: خواندن  
 منتقل میشود.



پاسخ:

در آغاز  $SCL=1$  که SDA صفر میشود و به معنی شروع ارتباط است، بعد از آن ۷ بیت آدرس اسلیو، بعد از آن یک بیت خواندن یا نوشت: خواندن یا نوشت را مشخص میکند، یک بیت Ack اسلیو، داده اصلی ۸ بیتی، یک بیت ack پایان ارتباط از مستر به اسلیو

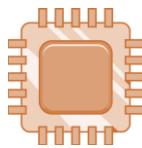
ب) در صورتی که بخواهیم ۱۰۰ بایت داده را از طریق دو رابط SPI و I2C ارسال کنیم، بازدهی (نسبت تعداد بیت داده به کل بیت ها) را در هر دو رابط محاسبه کنید. بازدهی کدام رابط بیشتر است؟

(فرض کنید در رابط SPI در هر frame ۱۶ بیت داده می توان ارسال کرد.)

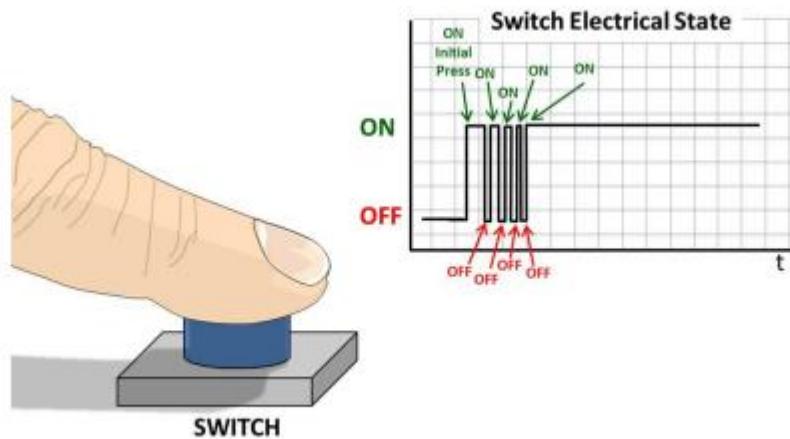
پاسخ:

SPI: در هر frame تمام ۱۶ بیت داده هستند، بنابراین به ازای هر تعداد بیت بازدهی برابر  $16/16 = 1$  است.

I2C: برای ارسال داده نیاز به ۱ بیت شروع + ۷ بیت آدرس + ۱ بیت W/R + ۱ بیت ACK + ۱ بیت داده است. به ازای هر بایت داده + ۱ بیت پایان داریم بنابراین برای ارسال ۱۰۰ بایت داده،  $100 * 1 = 100$  بیت باید مبادله شود بنابراین بازدهی برابر  $100/911 = 0.109$  می شود. در نتیجه بازدهی SPI بیشتر است.



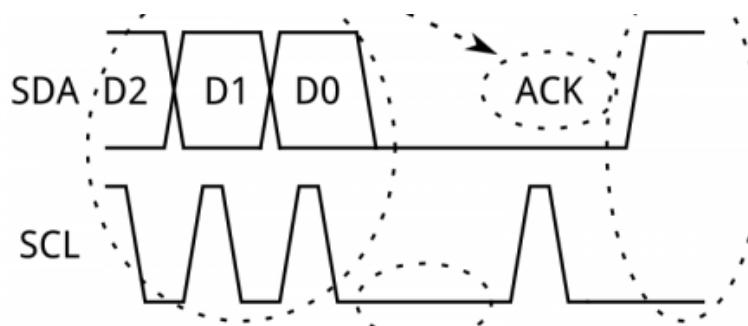
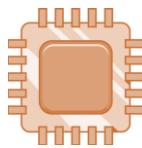
۶) الف) تصویر زیر پدیده‌ی Switch Bounce را نشان می‌دهد. نمونه‌ای از مشکلات احتمالی ناشی از این نوسان‌ها را بیان کنید و برای دوری از این مشکلات چه راه حلی وجود دارد؟ توضیح دهید.



پاسخ:

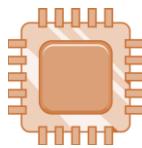
برای مثال بعد از فشردن دکمه بلند کردن صدا ممکن است صدا بیش از حد مطلوب بلند شود. زمانی که بخواهیم این نوسان‌ها را فیلتر کنیم از filter debouncing استفاده می‌شود. هنگامی که کلید مکانیکی فشرده یا رها می‌شود ممکن است خروجی برای مدتی نوسان کند، در برخی موارد این نوسان‌ها ممکن است منجر به رفتارهای ناخواسته و نامنتظره گردد. (برای نمونه ای جدی تر می‌توان به زمان فشار دادن دکمه افزایش سرعت تردیمیل اشاره کرد).

ب) در مورد مفهوم clock stretching تحقیق کنید.



: پاسخ

گاهی اوقات ممکن است بعضی از دستگاههای **slave** کلک را به اجبار پایین بکشند تا فرستادن اطلاعات بیشتر توسط **master** را به تأخیر بیاندازند یا تا زمان بیشتری برای آماده سازی داده قبل از این که آن را کلک بزند درخواست کنند. به این عمل **clock stretching** گفته میشود.



مهلت ارسال تمرین ساعت ۲۳:۵۵ روز دوشنبه ۱۷ آبان می‌باشد.

سوالات خود را می‌توانید از طریق تلگرام از تدریسیارهای گروه خود بپرسید.

ارائه پاسخ تمرین بهتر است به روش های زیر باشد:

۱) استفاده از فایل docx. تایپ پاسخها و ارائه فایل Pdf

۲) چاپ تمرین و پاسخ دهی به صورت دستنویس خوانا

فایل پاسخ تمرین را تنها با قالب **HW2 - 9731\*\*\*.pdf** در مدل بارگزاری کنید.

HW2- 9731097 نمونه:

فایل زیر ارسال نکنید.

11/8/2021



---

## Homework 3

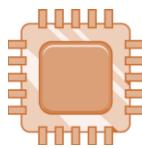
### Lec 9-12

---

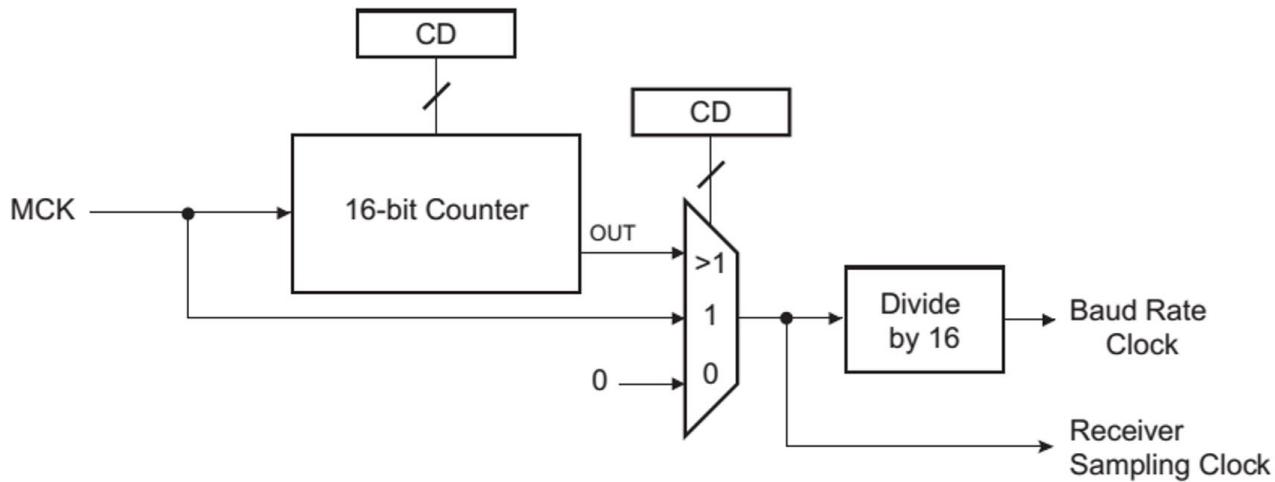


MICROPROCESSOR  
AND  
ASSEMBLY LANGUAGE

Fall 2021



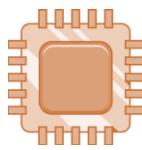
1) شکل زیر یک baud rate generator UART را نشان می‌دهد. به سوالات زیر در ارتباط با آن پاسخ دهید.



الف) با فرض این که  $MCK = 80MHz$  و  $CD = 8$  مقدار فرکانس گیرنده را در این صورت حساب کنید.

ب) با فرض این که  $MCK = 80MHz$  باشد مقدار حداقل و حداکثر baud rate چقدر خواهد بود.

ج) توضیح دهید چرا فرکانس گیرنده با فرکانس فرستنده تفاوت دارد.



# MICROPROCESSOR AND ASSEMBLY LANGUAGE

Dr. Farbeh

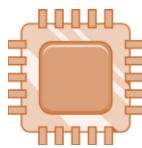
## Homework 3



2) در پروتکل UART مشخص کنید در موارد زیر کدام flag ها و رجیستر ها تغییر می کنند و تغییرات را ذکر کنید.

الف) تبادل اطلاعات بین فرستنده و گیرنده

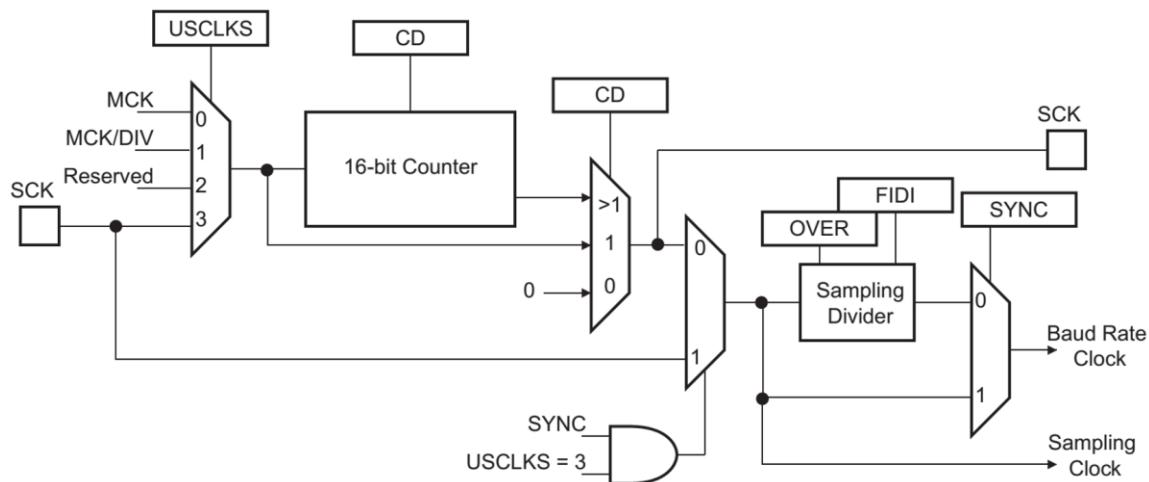
ب) در صورت سرازیر شدن رجیستر RHR



(3) به سوالات زیر در مورد baud rate generator برای ارتباط USART پاسخ دهید.

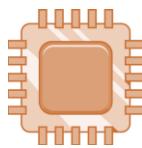
$MCK = 4\text{GHz}$

$MCK/\text{DIV} = 512\text{MHz}$



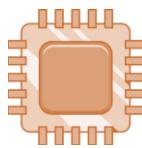
الف) اگر در حالت آسنکرون باشیم و در صورتی که baud rate = 4Kbps باشد رجیسترهاي CD, USCLKS, SYNC چه مقادیری باید داشته باشند.

ب) اگر در حالت سنکرون باشیم و فرض کنیم baud rate = 32Kbps باشد مقدار رجیسترهاي CD, USCLKS, SYNC چقدر باید باشد.

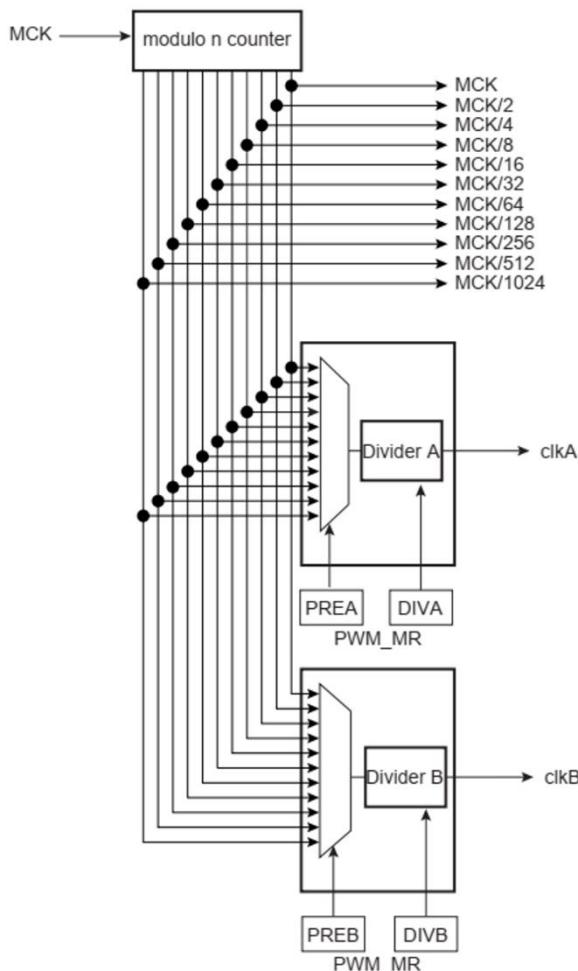


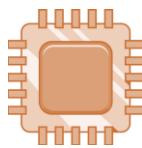
4) ورودی یک مبدل آنالوگ به دیجیتال ولتاژی در بازه  $[0\text{v}, 5\text{v}]$  را به اعداد 10 بیتی تبدیل می‌کند. اگر ورودی این مبدل از یک حسگر دما که بازه  $[-20^{\circ}\text{C}, 80^{\circ}\text{C}]$  را می‌تواند تشخیص دهد آمده باشد و دمای محیط  $30^{\circ}\text{C}$  باشد:

- الف) چه عددی به عنوان خروجی مبدل [D9-D0] نشان داده خواهد شد
- ب) عدد نشان داده شده در خروجی مبدل دقیقاً برابر چه دمایی است
- ج) علت تفاوت دمای خروجی مبدل با دمای اتاق چیست



5) میکروکنترلری با MCK = 500MHz در اختیار داریم با فرض این که PREA و DIVA ثبات‌های 16 بیتی باشند مقدار آن‌ها را طوری تنظیم کنید تا با اعمال حداقل فرکانس  $\text{clkA} = 1\text{KHz}$  شود.





6) پایه‌های USART را نام بده و کاربرد هر کدام را به اختصار توضیح دهید.

- مهلت ارسال تمرین ساعت 23:55 روز چهارشنبه 26 آبان می‌باشد.
- سوالات خود را می‌توانید از طریق تلگرام از تدریسیارهای گروه خود بپرسید.
- ارائه پاسخ تمرین به بهتر است به روش‌های 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 **HW3-9731\*\*\*.pdf** در مدل بارگزاری کنید.
- نمونه: HW3-9731590
- فایل زیپ ارسال نکنید.



11/8/2021



---

## Homework 3

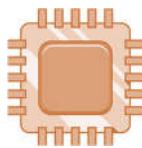
### Lec 9-12

---

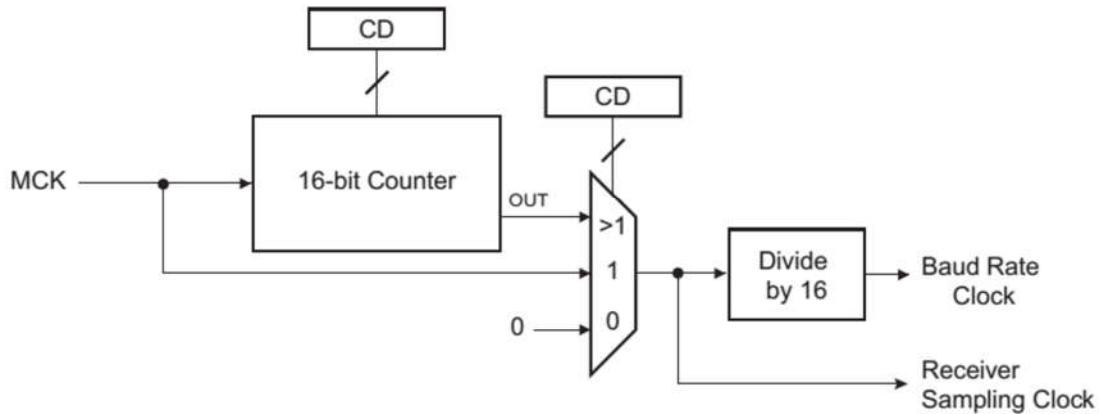


MICROPROCESSOR  
AND  
ASSEMBLY LANGUAGE

Fall 2021



(۱) شکل زیر یک baud rate generator UART را نشان می‌دهد. به سوالات زیر در ارتباط با آن پاسخ دهید.



الف) با فرض این که  $CD = 8$  مقدار فرکانس گیرنده را در این صورت حساب کنید.

ب) با فرض این که  $MCK = 80MHz$  باشد مقدار حداقل و حداکثر baud rate چقدر خواهد بود.

ج) توضیح دهید چرا فرکانس گیرنده با فرکانس فرستنده تفاوت دارد.

پاسخ:

(الف)

$$\text{baud rate} = \frac{MCK}{16 \times CD} = \frac{80MHz}{16 \times 8} = 625KHz$$

(ب)

حداکثر مقدار:  $CD=1$

$$\text{baud rate} = \frac{MCK}{16 \times CD} = \frac{80MHz}{16} = 5MHz$$

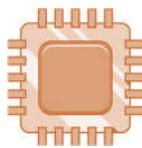
حداقل مقدار:  $CD = 65536$

$$\text{baud rate} = \frac{MCK}{16 \times CD} = \frac{80MHz}{16 \times 65536} \sim 77Hz$$

(ج)

فرکانس گیرنده ۱۶ برابر فرکانس فرستنده است از آنجایی که ممکن است روی خط ارتباطی نویز داشته باشد گیرنده فرکانس بالاتری دارد تا بتواند اطمینان حاصل کند که تغییر داده روی خط نویز است یا تغییر





(۲) در پروتکل UART مشخص کنید در موارد زیر کدام flag ها و رجیستر ها تغییر می‌کنند و تغییرات را ذکر کنید.

الف) تبادل اطلاعات بین فرستنده و گیرنده

ب) در صورت سرازیر شدن رجیستر RHR

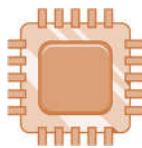
پاسخ:

الف)

یک بایت داده دریافت شده در SHIFT REGISTER ذخیره خواهد شد. صفر بودن فلگ RxRDY در STATUS REGISTER به معنای آماده بودن برای دریافت اطلاعات است هنگامی که یک بایت اطلاعات دریافت کنیم RxRDY برابر با یک خواهد شد و وقfe ای به پردازنده داده می‌شود و هشت بیت داده دریافت شده در رجیستر UART-RHR کپی خواهد شد. مقدار RxRDY در این مرحله صفر می‌شود و سپس SHIFT REGISTER خالی می‌شود و آماده دریافت داده های بعدی خواهد بود و مراحل قبل تکرار می‌شود

(ب)

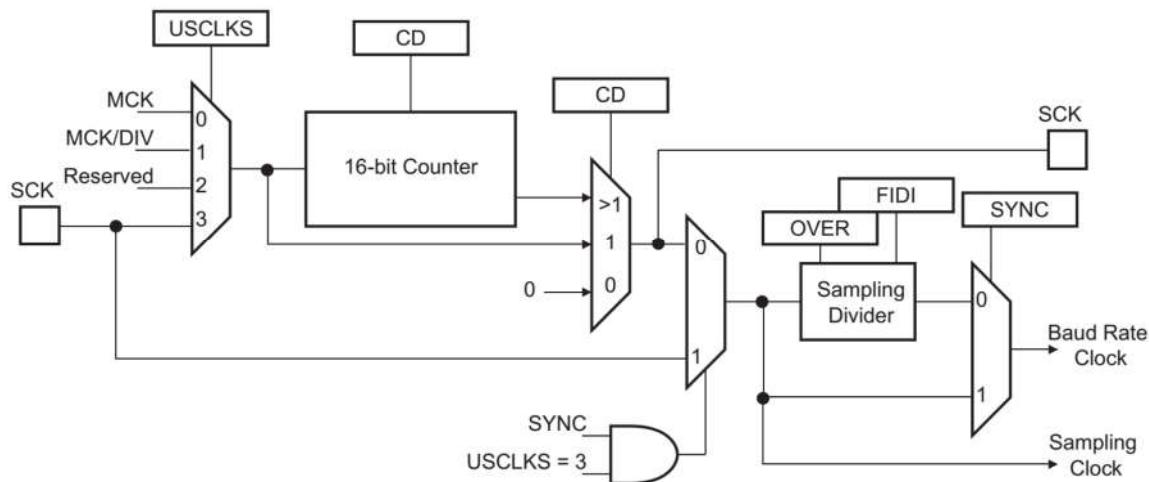
در صورتی که قبلاً از آمدن بایت بعدی RHR خالی نشود مشکل overrun خواهیم داشت. بیت OVER در رجیستر UART-SR یک می‌شود و به معنای سرریز شدن است و به سیستم ارور می‌فرستد. در این صورت برای برطرف کردن این مشکل باید فلگ RSTSTA یک شود



(۳) به سوالات زیر در مورد baud rate generator برای ارتباط USART پاسخ دهید.

$MCK = 4\text{GHz}$

$MCK/\text{DIV} = 512\text{MHz}$



الف) اگر در حالت آسنکرون باشیم و در صورتی که baud rate = 4Kbps باشد رجیسترهاي CD, USCLKS, SYNC چه مقادیری باید داشته باشند.

ب) اگر در حالت سنکرون باشیم و فرض کنیم baud rate = 32Kbps باشد مقدار رجیسترهاي CD, USCLKS, SYNC چقدر باید باشد.

پاسخ:

(الف)

SYNC = 0

USCLKS= 1

$$USCLKS = 0: \text{baud rate} = \frac{\frac{MCK}{\text{DIV}}}{8(2 - \text{OVER})\text{CD}} = \frac{512 \times 10^6}{8(2 - \text{OVER})\text{CD}} = 4 \times 10^3$$

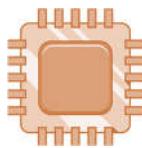
$$\rightarrow (2 - \text{OVER})\text{CD} = 32 \times 10^3$$

OVER = 0

CD = 16000

(ب)

SYNC = 1



حالت ۱: USCLKS = 0

$$USCLKS = 0: \text{baud rate} = \frac{MCK}{CD} = \frac{4 \times 10^9}{CD} = 32 \times 10^3 \rightarrow CD = 125K > 2^{16}$$

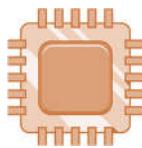
این حالت امکان پذیر نیست چرا که مقدار CD بزرگ تر از 16 بیت میشود.

حالت ۲: USCKLS = 1

$$USCLKS = 0: \text{baud rate} = \frac{MCK}{CD * DIV} = \frac{512 \times 10^6}{CD} = 32 \times 10^3 \rightarrow CD = 16K$$

پاسخ نهایی:

SYNC = 1, USCLKS = 1, CD=16000



(۴) ورودی یک مبدل آنالوگ به دیجیتال ولتاژی در بازه  $[0\text{v}, 5\text{v}]$  را به اعداد ۱۰ بیتی تبدیل می‌کند. اگر ورودی این مبدل از یک حسگر دما که بازه  $[-20^{\circ}\text{C}, 80^{\circ}\text{C}]$  را می‌تواند تشخیص دهد آمده باشد و دمای محیط  $30^{\circ}\text{C}$  باشد:

- الف) چه عددی به عنوان خروجی مبدل  $[D9-D0]$  نشان داده خواهد شد
- ب) عدد نشان داده شده در خروجی مبدل دقیقاً برابر چه دمایی است
- ج) علت تفاوت دمای خروجی مبدل با دمای اتاق چیست

پاسخ:

(الف)

$$\frac{30^{\circ} - (-20^{\circ})}{80^{\circ} - (-20^{\circ})} = \frac{V_{in} - 0}{5 - 0} \rightarrow V_{in} = 2.5v$$

$$N_{ADC} = 1023 \frac{V_{in} - 0}{5 - 0} = \frac{1023}{2} = 511.5 \xrightarrow{\text{Quantization Error}} N_{ADC} = 512$$

$$[D9 - D0] = 1000000000$$

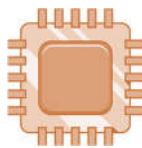
(ب)

$$V_{in} = 512 \frac{5 - 0}{1023} = 2.5024$$

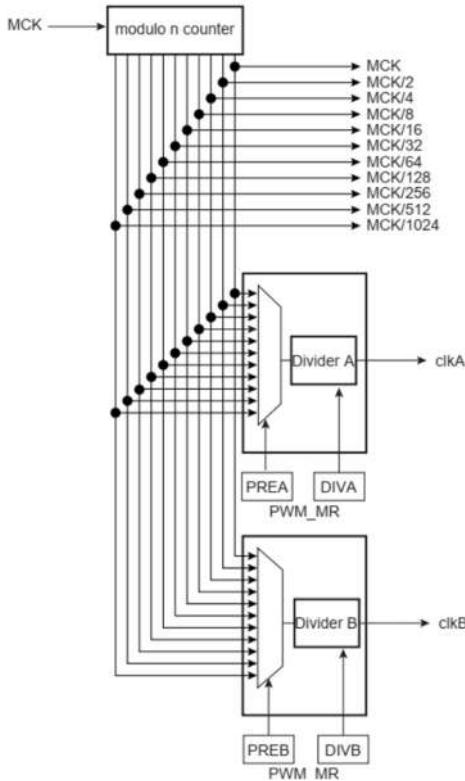
$$\frac{T - (-20^{\circ})}{80^{\circ} - (-20^{\circ})} = \frac{V_{in} - 0}{5 - 0} \rightarrow T + 20^{\circ} = 50.048 \rightarrow T = 30.048^{\circ}$$

(ج)

به دلیل Quantization Error



(۵) میکروکنترلری با اختیار داریم با فرض این که MCK = 500MHz در ثبات‌های DIVA و PREA بیتی باشند مقدار آن‌ها را طوری تنظیم کنید تا با اعمال حداقل فرکانس  $\text{clkA} = 1\text{KHz}$  شود.



پاسخ:

$$MCK = 500\text{MHz} \rightarrow clk = 1\text{KHz}$$

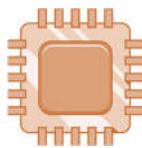
$$clk = \frac{MCK}{DIVA \times 2^{PREA}} = \frac{500 \times 10^6}{DIVA \times 2^{PREA}} = 10^3 \rightarrow DIVA \times 2^{PREA} = 5 \times 10^5$$

برای اعمال حداقل فرکانس باید ماکسیمم مقدار PREA را انتخاب کنیم.

در اینجا ماکسیمم مقدار  $PREA = 5$  است چرا که بیشتر از این مقدار دیگر DIVA یک عدد صحیح نخواهد بود.

$$PREA = 5 \rightarrow DIVA = \frac{5 \times 10^5}{32} = 15625$$





(۶) پایه‌های USART را نام بده و کاربرد هر کدام را به اختصار توضیح دهید.  
پاسخ:

۱- پایه SCK: پایه کلک برای حالت SYNC

۲- پایه TXD: پایه انتقال داده است اگر در مود SPI باشیم دو حالت دارد:

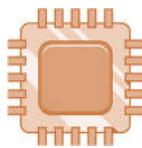
- در حالت مستر: MOSI
- در حالت اسلیو: MISO

۳- پایه RXD: برای دریافت اطلاعات است. اگر در مود SPI باشیم دو حالت دارد:

- در حالت مستر: MISO
- در حالت اسلیو: MOSI

۴- پایه CTS: پورت ورودی است. سیگنالی که فرستنده با آن می‌فهمد آمادگی دریافت داده را داریم. در حالت SPI در حالت اسلیو NSS است. این پایه active low می‌باشد

۵- پایه RTS: پورت خروجی است و از سمت گیرنده فعال می‌شود و اعلام می‌کند که گیرنده آمادگی دریافت دیتا را دارد این پورت به پورت CTS وصل می‌شود. پورت active low است و در حالت SPI در حالت مستر NSS است



- مهلت ارسال تمرین ساعت ۲۳,۵۵ روز یکشنبه هجدهم آبان می‌باشد.
- سوالات خود را می‌توانید از طریق تلگرام از تدریسیارهای گروه خود بپرسید.
- ارائه پاسخ تمرین به بهتر است به روش‌های زیر باشد:
  - (۱) استفاده از فایل .docx. تایپ پاسخها و ارائه فایل Pdf
  - (۲) چاپ تمرین و پاسخ دهی به صورت دستنویس خوانا
- فایل پاسخ تمرین را تنها با قالب **HW3-9731\*\*\*.pdf** در مدل بارگزاری کنید.
- نمونه: HW3-9731097
- فایل زیپ ارسال نکنید.

12/4/2021



---

## Homework 4

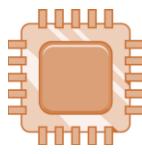
### Lec 13-18

---



MICROPROCESSOR  
AND  
ASSEMBLY LANGUAGE

Fall 2021



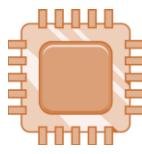
# MICROPROCESSOR AND ASSEMBLY LANGUAGE

Dr. Farbeh

## Homework 3



- 1) به سوالات زیر در مورد اسمنلر پاسخ دهید:
  - الف) اسمنلر و کامپایلر چه تفاوت و شباهتی باهم دارند؟
  - ب) با توجه به تفاوت‌های ذکر شده در قسمت الف اسمنلر چگونه Pseudo Instructions (شبه دستورات) را پیاده‌سازی می‌کند.



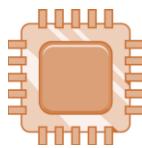
# MICROPROCESSOR AND ASSEMBLY LANGUAGE

Dr. Farbeh

## Homework 3



- 2) شباهت‌ها و تفاوت‌های سه مدل حافظه On chip ای که در میکرو درس وجود دارد را نام ببرید و به چه دلایلی برای ساخت میکروکنترل به این سه مدل حافظه نیاز داریم؟



(3) به سوالات زیر در مورد Directive ها توضیح دهید:

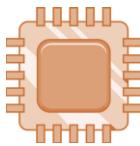
الف) Directive Area و انواع Attributes های آن را شرح دهید.

ب) چرا دو دستور زیر را در کنار هم در پایان برنامه های خود استفاده می کنیم و صرفا استفاده از Directive End به تنها بی کافی نیست؟

Here      B Here  
                End

ج) فرق بین سه Directive زیر چیست و از هر کدام برای چه کاربردی استفاده می شود(برای هر Directive یک مثال بزنید)?

Directives: DCB, SPACE, EQU



4) به سوالات زیر در مورد نگاشت حافظه پاسخ دهید:

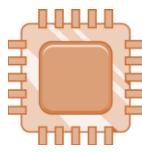
الف) در صورت اجرا برنامه زیر در نهایت در رजیستر R10 چه چیزی ذخیره می‌شود؟ (اعداد با متده Little در رجیسترها ذخیره می‌شوند).

```
Area Exercise4_Code, Readonly, Code
    LDR R2, =Our_Data;
    MOV R0, #9
    ADD R2, R2, R0;
    LDRB R10, [R2];
HERE    B HERE; stay here forever

Area Exercise4_Data, Data
Our_Data
    DCB "Micro_HW"
    DCD 0x50, 0x30
    END
```

ب) در صورتی که قبل از دستور DCD، دستور Align 4 اضافه کنیم نگاشت حافظه ما به چه صورت خواهد بود با فرض اینکه از خانه شماره صفر حافظه شروع به ذخیره دستورات کنیم؟ (همانطور که در ویدیوها مطرح شده، شبه دستور LDR نیز معادل با یک دستور اسembly خواهد بود و در یک 32 Bit سیو می‌شود).

ج) آیا امکان دارد بتوان شبه دستور LDR را با یک دستور دیگر جایگزین کرد؟



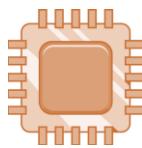
# MICROPROCESSOR AND ASSEMBLY LANGUAGE

Dr. Farbeh

## Homework 3



5) فرآیندی که پردازنده میکرو درس (Arm Cortex-M) بعد از شروع مجدد یا ری استارت برای شروع به کار طی می‌کند را شرح دهید.



6) با توجه به این نکته که طول دستورات در پردازنده Arm 32 Bit است چگونه آدرس خانه‌ای از حافظه که باید به آن Branch شود در این دستور ذخیره می‌شود.

- مهلت ارسال تمرین تا ساعت 23.55 روز سه شنبه بیست سه آذر می‌باشد.
- سوالات خود را می‌توانید از طریق تلگرام از تدریسیارهای گروه خود بپرسید.
- ارائه پاسخ تمرین به بهتر است به روش‌های 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 **HW4-9731\*\*\*.pdf** در مدل بارگزاری کنید.
- نمونه: HW4-9731121
- فایل زیپ ارسال نکنید.

12/4/2021



---

## Homework 4

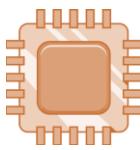
### Lec 13-18

---



MICROPROCESSOR  
AND  
ASSEMBLY LANGUAGE

Fall 2021



۱) به سوالات زیر در مورد اسمنلر پاسخ دهید:

الف) اسمنلر و کامپایلر چه تفاوت و شباهتی باهم دارند؟

پاسخ:

شباهت کامپایلر و اسمنلر در این مورد است که هردو کدی را به زبان ماشین تبدیل می‌کنند. یعنی یک دستور را ورودی می‌گیرند و در خروجی کد ۰۱ به ما می‌دهند.

نکته: در داخل کامپایلرهای اسمنلر وجود دارد و ابتدا دستور زبان‌های سطح بالا به اسمنلی تبدیل می‌شود و سپس توسط اسمنلر به ۰۱ تبدیل خواهد شد.

کامپایلر و اسمنلی در چند مورد زیر باهم تفاوت دارند:

۱- تفاوت اسمنلر و کامپایلر آن است که کامپایلر کد Level-High را به کد ماشین تبدیل می‌کند در صورتی که اسمنلر کد اسمنلی را به کد ماشین تبدیل می‌کند.

۲- تفاوت دیگر آن است که کامپایلر تمام کد را یکجا به زبان ماشین تبدیل می‌کند در صورتی که اسمنلر نمیتوانند این کار را انجام دهد و هر دستور به ترتیب به زبان ماشین تبدیل می‌شود.

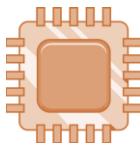
۳- کامپایلر باهوش‌تر از اسمنلر است زیرا فرایند Optimization بر روی تبدیل کد به زبان ماشین انجام می‌دهد اما اسمنلر صرفاً کد ماشین خروجی می‌دهد و بهینه سازی انجام نمی‌دهد. می‌توان گفت اسمنلر سریع‌تر است زیرا برخی فرایند‌های بهینه سازی کامپایلر را انجام نمی‌دهد.

۴- اسمنلر هر دستور اسمنلی به جز شبه دستورات رو به یک دستور سطح ماشین تبدیل می‌کند در حالی که کامپایلر لزوماً این کار را انجام نمی‌دهد و عموماً دستورات زبان‌های سطح بالا به چند دستور سطح ماشین تبدیل می‌شود.

ب) با توجه به تفاوت‌های ذکر شده در قسمت الف اسمنلر چگونه Pseudo Instructions (شبه دستورات) را پیاده‌سازی می‌کند.

پاسخ:

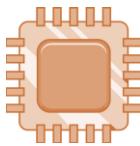
دستوراتی مانند LDR و ADR توسط اسمنلر به دستورات بیشتری شکسته می‌شوند (البته دستور LDR معادل یک دستور اسمنلی است). این دستورات برای پردازنده نیستند و صرفاً مختص به اسمنلر هستند که برای ساده‌تر شدن کار برنامه نویس طراحی شده و خود اسمنلر آنها را به چند دستور قبل فهم برای پردازنده تبدیل می‌کند. برای مثال در دستور MOV ماکریم یک عدد ۸ بیتی را می‌توان در رجیستر ذخیره کرد اما با شبه دستور LDR می‌توان اعداد بیشتر از ۸ بیت تا ۳۲ بیت را هم لود کرد.



۲) شباهت‌ها و تفاوت‌های سه مدل حافظه On chip ای که در میکرو درس وجود دارد را نام ببرید و به چه دلایلی برای ساخت میکروکنترل به این سه مدل حافظه نیاز داریم؟

پاسخ:

سه مدل حافظه‌ی گفته شده SRAM و EEPROM و FLASH می‌باشد.  
دو مدل حافظه‌ی EEPROM و FLASH حافظه‌های غیرفرار هستند. یعنی در صورت قطع شدن منبع انرژی، داده از حافظه پاک نمی‌شود. اما در سمت دیگر حافظه‌ی SRAM یک حافظه‌ی فرار یا VOLATILE می‌باشد و در صورت قطع جریان برق تمام داده‌های آن از بین می‌رود.  
حافظه‌ی EEPROM قابلیت WRITE کردن محدودی را دارد و پس از یک مقدار محدودی نوشتن بر روی آن دیگر قابلیت استفاده را ندارد. نحوه‌ی نوشتن بر روی FLASH و EEPROM متفاوت است. در فلش داده به صورت BLOCK نوشته می‌شود و در ابتدا باید داده‌ی قبلی یا همان BLOCK قبلی پاک شود و سپس بلوک جدید داده نوشته شود. در صورتی که در EEPROM نوشتن داده به صورت بایتی می‌باشد. به دلیل اینکه نوشتن بر روی فلش یک خرده دشوارتر است، داده‌هایی را بر روی آن می‌نویسیم که زیاد تغییر پیدا نمی‌کنند. مثال دستورالعمل‌ها را در فلش ذخیره می‌کنیم. و یا داده‌هایی که زیاد تغییر نمی‌کنند. همچنین یکی دیگر از دلایل استفاده از فلش میزان DENSITY یا تراکم آن می‌باشد چرا که میزان حجم حافظه‌ی بیشتری را به ما می‌دهد.  
در سمت دیگر داده را در EEPROM ذخیره می‌کنیم. چرا که خواندن و نوشتن بصورت بایتی می‌باشد. از لحاظ سرعت نیز SRAM دارای سرعت بیشتری نسبت به EEPROM می‌باشد. هم چنین میزان EEPROM محدودیت تعداد دفعات نوشتن ندارد.



(۳) به سوالات زیر در مورد Directive ها توضیح دهید:

الف) Directive Area و انواع Attributes های آن را شرح دهید.

پاسخ:

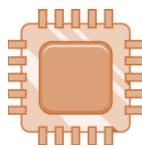
از این Directive برای بخش‌بندی کد و مرتب کردن و کمک به خوانایی آن استفاده می‌شود. یکی از این Attribute های Directive ReadOnly ویژگی Directive می‌باشد که مشخص می‌کند این قسمت از کد تنها قابل خواندن است و بعد از ذخیره شدن دیگر تغییر نمی‌کند و این بخش در Flash ذخیره خواهد شد. در مقابل ReadOnly ما قابلیت ReadWrite را داریم که مشخص می‌کند این بخش قابلیت خواندن و نوشتمن دارد و در SRam ذخیره خواهد شد. یکی دیگر از Attribute های آن Code می‌باشد که مشخص می‌کند این قسمت مربوط به کد برنامه می‌باشد؛ این ویژگی، ReadOnly نیز می‌باشد. یعنی اگر از Code استفاده کنیم، آن قسمت تنها قابل خواندن خواهد بود. یکی دیگر از Directive های این Attribute DATA می‌باشد و مشخص می‌کند که این قسمت مربوط به داده‌ی برنامه می‌باشد. این قابلیت ReadWrite را به این قسمت از کد می‌دهد. آخرین ویژگی این Directive Align ویژگی Align است. این Attribute مشخص می‌کند که داده به صورت Align شده (یعنی مثلاً از ضرایب ۲ یا ۴ در حافظه ذخیره شود) در حافظه ذخیره شود.

ب) چرا دو دستور زیر را در کنارهم در پایان برنامه‌های خود استفاده می‌کنیم و صرفا استفاده از Directive End به تنها‌یابی کافی نیست؟

Here      B Here  
End

پاسخ:

صرفه برای اسمبلر کاربرد دارد و مشخص می‌کند بعد این خط دیگر جزو دستورات برنامه Directive End ما نیست و اسمبلر نباید آنها را به زبان ماشین تبدیل کند. در حالی که در صورت نبودن HERE B و پس از اجرا شدن دستورالعمل ها، PC دوباره افزایش پیدا می‌کند و خانه‌های دیگری از حافظه را نیز به عنوان دستور العمل تعبیر می‌کند و آنها را fetch می‌کند و به عنوان دستورالعمل اجرا می‌کند. این کار که ناخواسته می‌باشد موجب اجرا شدن دستورالعمل‌های invalid یا valid می‌شود. برای جلوگیری از این کار از branch و لیبل HERE استفاده می‌کنیم. یعنی یک حلقه به وجود می‌آید و HERE یک لیبل هست که به خانه‌ی فعلی حافظه اشاره می‌کند. پس این دستور موجب می‌شود که در این خانه از حافظه تا ابد! بماند.



ج) فرق بین سه زیر چیست و از هر کدام برای چه کاربردی استفاده می‌شود(برای هر یک مثال بزنید)? Directive

### Directives: DCB, SPACE, EQU

پاسخ:

EQU: برای تعریف یک Constant در برنامه خود از EQU استفاده می‌کنیم و اسمبلر در هر جای برنامه مانند مثال اگر عبارت Count را ببینند آن را به 0x25 تبدیل می‌کند.

Count EQU 0x25

DCB: بعد از اینکه اسمبلر به این خط می‌رسد به اندازه یک بایت بعد از آن دستور در حافظه فضا رزرو می‌کند و برای آن اسم مشخص شده را قرار می‌دهد. باید حتماً مقدار اولیه مشخص باشد.

Myvalue DCB 5

SPACE: به اندازه حافظه مشخص شده از فضای حافظه ذخیره می‌کند و اسم داده شده را ببروی آن می‌گذارد. نیازی نیست به آن مقدار اولیه بدهیم.

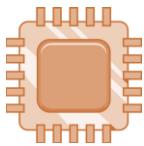
Long\_var Space 4

۴) به سوالات زیر در مورد نگاشت حافظه پاسخ دهید:

الف) در صورت اجرا برنامه زیر در نهایت در رجیستر R10 چه چیزی ذخیره می‌شود؟ (اعداد با متدهای LittleEndian در رجیسترها ذخیره می‌شوند).

```
Area Exercise4_Code, Readonly, Code
    LDR R2, =Our_Data;
    MOV R0, #9
    ADD R2, R2, R0;
    LDRB R10, [R2];
HERE    B HERE; stay here forever

Area Exercise4_Data, Data
Our_Data
    DCB    "Micro_HW"
    DCD    0x50, 0x30
    END
```



پاسخ:

ب) با توجه به اینکه آدرسی که Our\_Data به آن اشاره می‌کند در ابتدا همان آدرس باقی است که کاراکتر m در آن ذخیره شده پس بعد از اضافه کردن #9 به مقدار آن به بایت دوم پر ارزش از Word ای که در آن little endian Word 0x50 ذخیره شده است اشاره خواهد کرد و چون مقدار 0x50 در آن Word به صورت ذخیره شده است مقدار بایت دوم پر ارزش صفر خواهد بود و در نهایت مقدار صفر در R10 لود می‌شود.

ب) در صورتی که قبل از دستور DCD، دستور Align 4 اضافه کنیم نگاشت حافظه ما به چه صورت خواهد بود با فرض اینکه از خانه شماره صفر حافظه شروع به ذخیره دستورات کنیم؟ (همانطور که در ویدیوها مطرح شده، شبیه دستور LDR نیز معادل با یک دستور اسمبلی خواهد بود و در یک 32 Bit سیو می‌شود).

پاسخ:

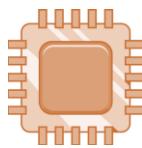
با توجه به اینکه همه دستورات ما در 4 بایت ذخیره می‌شوند و حتی رشته ما در 8 بایت (هر کاراکتر به 1 بایت برای ذخیره شدن نیاز دارد). سیو می‌شود و در نهایت دستور DCD یک 4 بایتی را رزرو می‌کند، در نتیجه Align 4 تاثیری در نگاشت حافظه ما ایجاد نمی‌کند و مشابه حالت قبل خواهد بود.

ج) آیا امکان دارد بتوان شبیه دستور LDR را با یک دستور دیگر جایگزین کرد؟

پاسخ:

بله، می‌توانیم از شبیه دستور ADR استفاده کنیم که به صورت زیر خواهد شد دستور معادل ADR R2, Our\_Data

نکته: در صورت استفاده از شبیه دستور ADR دیگر نیازی نیست از = استفاده کنیم.



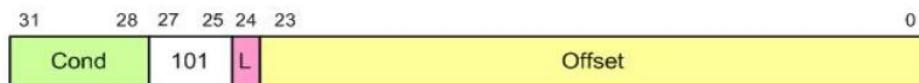
۵) فرآیندی که پردازنده میکرو درس (Arm Cortex-M) بعد از شروع مجدد یا ری استارت برای شروع به کار طی می‌کند را شرح دهید.

پاسخ:

برخلاف باقی پردازنده‌های شرکت Arm که از خانه صفر شروع به خواندن دستورات از حافظه می‌کنند، در Arm Cortex-M ابتدا پردازنده از خانه شماره ۴ تا ۷ حافظه را می‌خواند که در این ۳۲ بیت آدرسی ذخیره شده است که باید PC از آن شروع کند. در نتیجه این مقدار را در PC ذخیره می‌کند و به سراغ اجرای برنامه خود می‌رود. پس بعد از ذخیره برنامه خود در میکرو باید ادرس دستور اول برنامه خود را در خانه شماره ۴ تا ۷ حافظه ذخیره کنیم تا میکرو به درستی Program شود.

۶) با توجه به این نکته که طول دستورات در پردازنده Arm 32 Bit است چگونه آدرس خانه‌ای از حافظه که باید به آن Branch شود در این دستور ذخیره می‌شود.

پاسخ:



همانطور که مشاهده می‌کنید ساختار دستورات Branch به شکل بالا است و تنها می‌توان یک آدرس ۲۴ بیتی را در بخش Offset ذخیره کرد در نتیجه ما با استفاده از دستور Branch به تمامی حافظه دسترسی نداریم و تنها به بخشی از آن دسترسی خواهیم داشت. Arm برای اینکه ما بتوانیم به خانه‌های بیشتری از حافظه دسترسی داشته باشیم نه صرفا خانه‌هایی از حافظه که آدرس آن‌ها ۲۴ بیت یا کمتر است، امکانی را ایجاد کرده که این بخش Offset با آدرس کنونی ما جمع می‌شود و این مدلی این آدرس ۲۴ بیتی فاصله ما به خانه حافظه Branch را تعیین می‌کند نه آدرس مطلق آن خانه حافظه و ما می‌توانیم به بخش بیشتری از حافظه دسترسی داشته باشیم.

12/14/2021



---

## Homework 5

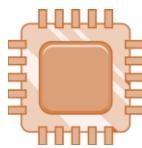
### Lec 19-21

---



MICROPROCESSOR  
AND  
ASSEMBLY LANGUAGE

Fall 2021



۱) برنامه‌ای بنویسید که با استفاده از آن بتوان تشخیص داد که مقدار قرارگرفته در رجیستر R0 پالیندروم است یا خیر. (برای مثال، ۱۱۰۰ یک پالیندروم ۴ بیتی است).

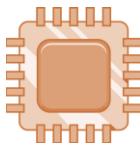
۲) هنگامی که بر روی کیبورد، دو کاراکتر ۴ و ۶ را تایپ می‌کنیم، ۰x36 و ۰x34 در واقع به ما داده می‌شود. برنامه‌ای بنویسید که ۰x34 و ۰x36 را به packed BCD تبدیل کرده و نتیجه را در رجیستر R2 ذخیره نماید.

۳) الف) کد اsemblی معادل قطعه کد زیر را بنویسید. (مقادیر متغیرهای استفاده شده در ثبات‌ها طبق جدول زیر ذخیره شده‌است).

a	R4
b	R5
c	R6

```
while (a - b > 0) {
    if (a > -b) {
        c = c - a;
        a = -a;
    }
    else {
        b = c * b;
        a = 2 - b;
    }
}
```

ب) اگر قصد داشته باشیم کد if ((R0==R1) && (R2==R3)): R4++ را به زبان اsemblی بنویسیم، کد مناسب را فقط با سه دستور پیاده کنید.



۴) برنامه‌ای بنویسید که مقدار  $b$  م و  $c$  م دو مقدار ذخیره شده در  $R0$  و  $R1$  را محاسبه کرده و به ترتیب در  $R2$  و  $R3$  ذخیره کند.

۵) برنامه‌ای بنویسید که مقدار ذخیره شده در ثبات  $R0$  را در یک آرایه ۱۰ عضوی به روش دودویی، جستجو کند (binary search) (فرض کنید که آرایه از قبل به صورت صعودی، مرتب شده است).

۶) برنامه‌ای بنویسید که جمله  $n$  ام دنباله فیبوناچی را در ثبات  $R1$  قرار دهد (مقدار  $n$  در ثبات  $R0$  قرار گرفته است).

• مهلت ارسال تمرین ساعت ۲۳.۵۵ روز جمعه سوم دیماه می‌باشد.

• سوالات خود را می‌توانید از طریق تلگرام از تدریسیارهای گروه خود بپرسید.

• کدهای اسمنبلی را با استفاده از keil انجام دهید.

• ارائه پاسخ تمرین به بهتر است به روش‌های زیر باشد:

۱) ارائه اسکرین شات از کد و نتیجه اجرای آن در یک فایل pdf

۲) قرار دادن فایل کد و اسکرین شات از نتیجه اجرای کد. در صورت استفاده از این روش حتما هر سوال را در پوشه جداگانه قرار دهید.

• فایل پاسخ تمرین را تنها با قالب **HW5 - 9731\*\*\*.pdf** یا **HW5 - 9731\*\*\*.zip** در مودل بارگذاری کنید.

• نمونه: HW5-9731097

12/14/2021



---

## Homework Solution 5

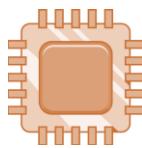
### Lec 19-21

---



MICROPROCESSOR  
AND  
ASSEMBLY LANGUAGE

Fall 2021



۱) برنامه‌ای بنویسید که با استفاده از آن بتوان تشخیص داد که مقدار قرارگرفته در رجیستر R0 پالیندروم است یا خیر. (برای مثال، ۱۱۰ یک پالیندروم ۴ بیتی است).

ابتدا بایستی مقدار قرار گرفته در R0 reverse و سپس آن را با مقدار اولیه داده شده در ثبات R0 مقایسه کنیم. اگر دو مقدار برابر باشند، پالیندروم داریم.

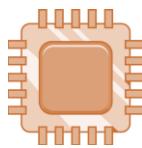
```
AREA my_data, DATA
INPUT EQU 0x80000001 ;change value for different inputs

EXPORT _main
AREA palindrome, CODE, READONLY
ENTRY

_main
    LDR r0,=INPUT
    MOV r1,r0 ; make copy of input
    MOV r2,#0 ;used for storing inverted number
    MOV r3,#32 ; loop counter
    MOV r4,#0 ; is the result

loop
    AND r5,r1,#1 ;take only the first bit
    LSL r2,r2,#1 ; shift one to left
    ADD r2,r2,r5
    LSR r1,r1,#1
    SUBS r3,r3,#1
    BNE loop

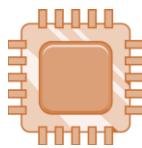
    CMP r0,r2
    BNE here
    MOV r4,#1
here b here
END
```



۲) هنگامی که بر روی کیبورد، دو کاراکتر ۴ و ۶ را تایپ می‌کنیم، ۰x34 و ۰x36 در واقع به ما داده می‌شود. برنامه‌ای بنویسید که ۰x34 و ۰x36 را به packed BCD تبدیل کرده و نتیجه را در رجیستر R2 ذخیره نماید.

```
EXPORT _main
AREA bcd, CODE, READONLY
ENTRY

_main
    MOV R0, 0x34 ; input 4
    MOV R1, 0x36 ; input 6
    MOV R3, 0xF ; used for AND to make unpacked BCD
    ;turn r0 and r1 to bcd of each
    AND R0, R0, R3
    AND R1, R1, R3
    ;combine the two BCD values in one register
    LSL R2, R0, #4
    ADD R2, R1
here b here
END
```

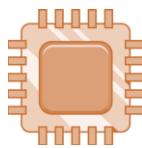


۳) الف) کد اسembly معادل قطعه کد زیر را بنویسید. (مقادیر متغیرهای استفاده شده در ثباتها طبق جدول زیر ذخیره شده است).

a	R4
b	R5
c	R6

```
while (a - b > 0) {  
    if (a > -b) {  
        c = c - a;  
        a = -a;  
    }  
    else {  
        b = c * b;  
        a = 2 - b;  
    }  
}
```

```
export _main  
area start ,code, readonly  
entry  
  
_main  
    b while  
while  
    neg r7,r5  
    cmp r4,r5  
    bhi ifelse  
  
ifelse  
    cmp r4 ,r7  
    bhi iff  
    mul r5,r5,r6  
    rsb r5,r5,#2  
iff  
    sub r6,r6,r4  
    neg r4,r4  
    b while  
  
end
```



ب) اگر قصد داشته باشیم کد if ((R0==R1) && (R2==R3)): R4++ را به زبان اسembly بنویسیم،  
کد مناسب را فقط با سه دستور پیاده کنید.

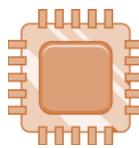
---

AREA myData, DATA

EXPORT \_\_main  
AREA myCode, CODE, READONLY  
ENTRY

\_\_main  
    CMP r0, r1;  
    CMPEQ r2, r3;  
    ADD r4, #1;

HERE B HERE; stay here forever



# MICROPROCESSOR AND ASSEMBLY LANGUAGE

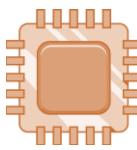
Dr. Farbeh

## Homework 5



۴) برنامه‌ای بنویسید که مقدار ب م و ک م دو مقدار ذخیره شده در R0 و R1 را محاسبه کرده و به ترتیب در R2 و R3 ذخیره کند.

```
1      AREA myData, DATA
2      CONST_1 EQU 0x34
3
4      num1 RN r0;
5      num2 RN r1;
6
7      n1 RN r2; temp numbers that we work them for gcd
8      n2 RN r3;
9
10
11     GCD RN r4;
12     LCM RN r5;
13
14     TMP RN r6;
15
16
17     EXPORT __main
18     AREA myCode, CODE, READONLY
19     ENTRY
20
21     __main
22
23         LDR num1, =54;
24         LDR num2, =13;
25         MOV n1, num1;
26         MOV n2, num2;
27
28     loop
29
30         CMP n1,n2;
31         BNE inside_if_else;
32         MOV GCD, n1;
33         B inside_lcm; end the loop if n1 == n2
34
35     inside_if_else
36         CMP n1, n2;
37         BLS inside_else;
38
39         SUB n1, n1, n2;inside if
40         B loop;
41     inside_else
42         SUB n2, n2, n1;inside else
43         B loop;
44
45     inside_lcm ;this line calculate the lcm
46         MUL TMP, num1, num2;
47         UDIV LCM, TMP, GCD;
48
49     HERE B HERE;
50     END
```



# MICROPROCESSOR AND ASSEMBLY LANGUAGE

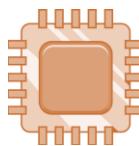
Dr. Farbeh

## Homework 5



۵) برنامه‌ای بنویسید که مقدار ذخیره شده در ثبات R0 را در یک آرایه ۱۰ عضوی به روش دودویی، جستجو کند (binary search) (فرض کنید که آرایه از قبل به صورت صعودی، مرتب شده است.)

```
1      AREA myData, DATA, READWRITE
2      value_to_find_const EQU 9;
3      mem_addr RN r4;
4      mem_val RN r5;
5      left RN r6;
6      right RN r7;
7      mid RN r8;
8      tmp RN r9;
9      value_to_find RN r10;
10     found RN r11;
11     arr_size EQU 10;
12     EXPORT __main
13     AREA myCode, CODE, READONLY
14     ENTRY
15
16     __main
17     LDR mem_addr, =ARR;
18     LDR left, =0;
19     LDR right, =arr_size;
20     SUB right,right,#1;
21     LDR mid, =0;
22     LDR value_to_find, =value_to_find_const;
23     LDR found, =0;
24
25     loop
26     CMP left, right; check to loop condition here
27     BLS inside_loop;
28     B here;|
29     inside_loop
30     ADD mid, left, right;
31     LDR tmp, =2;
32     UDIV mid, mid, tmp; calculating the mid -> mid=(left+right)/2
33
34     LDR tmp,=ARR;
35     ADD tmp, mid, tmp;
36     LDRB mem_val, [tmp]; load just one byte
37
38     CMP mem_val, value_to_find;
39     BNE else_if;
40     LDR found, =1;
41     B here;
42
43     else_if
44     BHI inside_else;
45     ADD left, mid, #1;
46     B loop;
47
48     inside_else
49     SUB right, mid, #1;
50     B loop;
51
52     ALIGN 4;
53     ARR DCB 1,2,2,4,5,6,7,9,11,13;
54
55     here B here;
56
57     END
|
```



۶) برنامه‌ای بنویسید که جمله  $n$  ام دنبالهٔ فیبوناچی را در ثبات  $R1$  قرار دهد (مقدار  $n$  در ثبات  $R0$  گرفته است).

```
1      AREA myData, DATA, READWRITE
2
3      Nth  RN r0;
4      fib_val RN r1;
5
6      TMP_1 RN r2;
7      TMP_2 RN r3;
8      TMP_3 RN r4;
9
10
11     EXPORT __main
12     AREA myCode, CODE, READONLY
13     ENTRY
14
15     __main
16     LDR Nth,=10; 5th member of fib
17     LDR TMP_1, =0;
18     LDR TMP_2, =1;
19     LDR fib_val, =0;
20     CMP Nth,#1;
21     BHI loop;if higher than one
22
23     BNE here;
24     LDR fib_val, = 1;
25     B here;[  
loop
26     SUB Nth, Nth, #2;
27
28     inside_loop
29     MOV TMP_3, TMP_2;
30     ADD TMP_2, TMP_1, TMP_2;
31     MOV TMP_1, TMP_3;
32
33     SUBS Nth, #1;
34     BNE inside_loop;
35
36     MOV fib_val,TMP_2;
37 here B here;
38
39     END
40
41 |
```

12/25/2020



---

## Homework 6

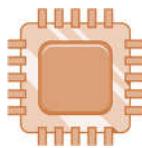
### Lec 22-25

---



MICROPROCESSOR  
AND  
ASSEMBLY LANGUAGE

Fall 2021



۱) فرض کنید وضعیت حافظه و رجیسترها به شکل زیر باشد:

آدرس حافظه

0x8010	0x00000001
0x800C	0xFFEEDDEA
0x8008	0x00008888
0x8004	0x12340000
0x8000	0xBAE00000

رجیستر

0x13	R0
0xFFFFFFFF	R1
0xEEEEEEEE	R2
0x8000	R3

پس از اجرای دستور زیر وضعیت و محتوای حافظه و رجیسترها بکشید و دلیل آن را توضیح دهید.

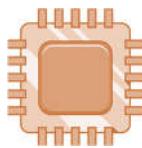
LDMIA R3!, {R0, R1, R2}

۲) برنامه ای به زبان اسمنبلی بنویسید که نشان دهد یک عدد دلخواه اول است یا خیر. (عدد دلخواه را در رجیستر R0 قرار دهید. همچنین برای مشخص کردن اول نبودن عدد 0x00000000 و برای اول بودن عدد 0x11111111 را در رجیستر R3 بریزید)

۳) عددی را در خانه 0x05000000 ثبت کنید. برنامه ای بنویسید که آن را تقسیم بر توان های 2، از یک تا ده کند و آن را در ده رجیستر اول بریزد

۴) قطعه کد اسمنبلی معادل با کد C زیر را بنویسید  
(الف)

```
for (R0 = 0 ; R0 < 10; R0++){
    if (R1 == 0) {
        R2++;
    }
}
```



(ب)

```
int *ptr;  
int sum = 0;  
for (int i = 0; i < 20; i++)  
    sum += *(ptr++);
```

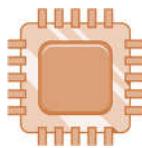
۵) در هر بخش آدرس خانه حافظه‌ای که به آن اشاره می‌شود را بدست آورید و رجیستری که با علامت سوال مشخص شده است را در هر مورد بنویسید. مراحل کار خود را توضیح دهید.

در همه موارد فرض کنید: R5=0x4000, R4=0x20

همچنین هرجا نیاز به خواندن خانه‌ای از حافظه را داشتید مقدار آن را 0xFF فرض کنید.

- a. LDR R9, =0x11223344  
STRH R9, [R3, R4]
- b. LDRB R8, [R3, R4, LSL #3] ;R8 = ?
- c. LDR R7, [R3], R4 ;R7 = ?, R3 = ?
- d. LDR R6, =0x11223344  
STRB R6, [R3], R4, ASR #2, R3 = ?

۶) ۴ مورد از قوانین استاندارد AAPCS برای پیاده‌سازی توابع را نام ببرید



- مهلت ارسال تمرین ساعت ۲۳.۵۵ روز جمعه هفدهم دی ماه است
- سوالات خود را می‌توانید از طریق تلگرام از تدریسیاران گروه خود بپرسید
- کد های اسembly را با استفاده از keil انجام دهید
- ارائه پاسخ تمرین بهتر است به روش های زیر انجام شود.
  - (۱) ارائه اسکرین شات از کد و نتیجه اجرای آن در یک فایل pdf
  - (۲) قرار دادن فایل کد و اسکرین شات از نتیجه اجرای کد. در صورت استفاده از این روش حتما هر سوال را در پوشه جداگانه قرار دهید.
- فایل پاسخ تمرین را تنها با قالب **HW6-9731\*\*\*.pdf** یا **HW6-9731\*\*\*.zip** در مدل بارگزاری کنید.
- HW6-9731097.pdf

12/25/2020



---

## Homework 6

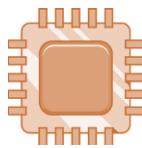
### Lec 22-25

---



MICROPROCESSOR  
AND  
ASSEMBLY LANGUAGE

Fall 2021



1) فرض کنید وضعیت حافظه و رجیسترها به شکل زیر باشد:

آدرس حافظه

0x8010	0x00000001
0x800C	0xFEEDDEAF
0x8008	0x00008888
0x8004	0x12340000
0x8000	0xBABE0000

رجیستر

0x13	R0
0xFFFFFFFF	R1
0xEEEEEEEE	R2
0x8000	R3

پس از اجرای دستور زیر وضعیت و محتوای حافظه و رجیسترها بکشید و دلیل آن را توضیح دهید.

LDMIA R3!, {R0, R1, R2}

پاسخ:

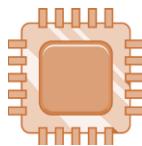
در اینجا دستور LDM به صورت Increasing After انجام می‌شود در نتیجه پس از هربار خواندن از مموری به اندازه ۴ واحد پوینتر افزایش می‌یابد و در انتها نیز مقدار نهایی پوینتر در رجیستر R3 ذخیره می‌شود. مقادیر خوانده شده از حافظه نیز به ترتیب در R2, R1, R0 ذخیره می‌شوند.

آدرس حافظه

0x8010	0x00000001
0x800C	0xFEEDDEAF
0x8008	0x00008888
0x8004	0x12340000
0x8000	0xBABE0000

رجیستر

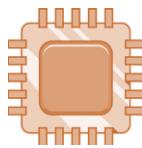
0xBABE0000	R0
0x12340000	R1
0x00008888	R2
0x800C	R3



2) برنامه ای به زبان اسembly بنویسید که نشان دهد یک عدد دلخواه اول است یا خیر. (عدد دلخواه را در رجیستر R0 قرار دهید. همچنین برای مشخص کردن اول نبودن عدد 0x00000000 و برای اول بودن عدد 0x11111111 را در رجیستر R3 بروزیزد)

: پاسخ

```
AREA Prime_or_Not,code,readonly
ENTRY
    MOV R0,#15;           Number which you want to test
    CMP R0,#01;           Comparing with 01
    BEQ PRIME;           If equal declare directly as prime
    CMP R0,#02;           Compare with 02
    BEQ PRIME;           If equal declare directly as prime
    MOV R1,R0;            Copy test number in R1
    MOV R2,#02;           Initial divider
    UP
    BL DIVISION;         Call for division sub-function
    CMP R8,#0;            Compare remainder with 0
    BEQ NOTPRIME;         If equal then its not prime
    ADD R2,R2,#01;         If not increment divider and check
    CMP R2,R1;            Compare divider with test number
    BEQ PRIME;            All possible numbers are done means It's
prime
    B UP;                 If not repeat until end
NOTPRIME
    LDR R3,=0x11111111;   Declaring test number is not prime
    B STOP;                Jumping to infinite looping
PRIME
    LDR R3,=0xFFFFFFFF;   Declaring test number is prime
number
    STOP B STOP;           Infinite looping
```



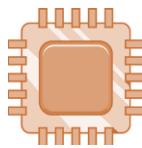
# MICROPROCESSOR AND ASSEMBLY LANGUAGE

Dr. Farbeh

## Homework 6



DIVISION;	Function for division operation
MOV R8,R0;	Copy of data from main function
MOV R9,R2;	Copy of divider from main function
LOOP	
SUB R8,R8,R9;	Successive subtraction for division
ADD R10,R10,#01;	Counter for holding the result of division
CMP R8,R9;	Compares for non-zero result
BPL LOOP;	Repeats the loop if subtraction is still needed
MOV PC,LR;	Return back to main function
END	



3) عددی را در خانه 0x05000000 ثبت کنید. برنامه ای بنویسید که آن را تقسیم بر توان های 2، از یک تا ده کند و آن را در ده رجیستر اول بریزد

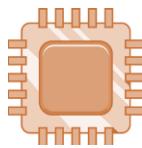
: پاسخ

```
LDR r0, =4096
LDR r1, =0x50000000
STR r0, [r1]

LDR r10, [r1]

ASR r0, r10, #1
ASR r1, r0, #1
ASR r2, r1, #1
ASR r3, r2, #1
ASR r4, r3, #1
ASR r5, r4, #1
ASR r6, r5, #1
ASR r7, r6, #1
ASR r8, r7, #1
ASR r9, r8, #1

here B here
```



4) قطعه کد اسambilی معادل با کد C زیر را بنویسید

(الف)

```
for (R0 = 0 ; R0 < 10; R0++){
    if (R1 == 0) {
        R2++;
    }
}
```

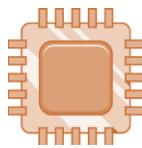
(ب)

```
int *ptr;
int sum = 0;
for (int i = 0; i < 20; i++)
    sum += *(ptr++);
```

: پاسخ

(الف)

```
MOV R0, #0
AGAIN CMP R0, #10
BEQ HERE
CMP R1, #0
ADDEQ R2, R2, #1
ADD R0, R0, #1
B AGAIN
HERE B HERE END
```



# MICROPROCESSOR AND ASSEMBLY LANGUAGE

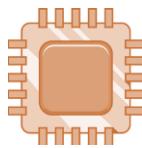
Dr. Farbeh

## Homework 6



(ب)

```
ADR R0, 0X400
MOV R1, #0
MOV R2, #21
LOOP CMP R2, #1
BEQ FINISH
ADD R0, R0, #4
LDR R3, [R0]
ADD R1, R1, R3
SUB R2, R2, #1
B LOOP
FINISH END
```



5) در هر بخش آدرس خانه حافظه‌ای که به آن اشاره می‌شود را بدست آورید و رجیستری که با علامت سوال مشخص شده است را در هر مورد بنویسید. مراحل کار خود را توضیح دهید.

در همه موارد فرض کنید:  $R3=0x4000$ ,  $R4=0x20$

همچنین هرجا نیاز به خواندن خانه‌ای از حافظه را داشتید مقدار آن را  $0xFF$  فرض کنید.

a.  $LDR R9, =0x11223344$   
 $STRH R9, [R3, R4]$

b.  $LDRB R8, [R3, R4, LSL #3] ; R8 = ?$

c.  $LDR R7, [R3], R4 ; R7 = ?, R3 = ?$

d.  $LDR R6, =0x11223344$   
 $STRB R6, [R3], R4, ASR #2, R3 = ?$

پاسخ:

\*در صورت سوال اشتباهی به جای  $R3$ ,  $R5$ ,  $R3$  قرار داده شده است. در صورتی که  $R3$  را به صورت فرضی در نظر گرفته باشید باز نیز مورد قبول است.

(a) نوع آدرس دهی در این بخش: Pre-indexed addressing mode with fixed offset

در این حالت مقدار  $R4$  با  $R3$  جمع می‌شود و  $R9$  در آدرس حاصل جمع ذخیره می‌شود:

$$R3 + R4 = 0x4020$$

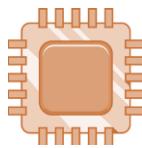
$$\text{Memory Address } 0x4020 = 0x44$$

$$\text{Memory Address } 0x4021 = 0x33$$

(b) نوع آدرس دهی در این بخش: Pre-indexed addressing mode with fixed offset of shifted register

در این حالت مقدار  $R4$  سه بار به چپ شیفت منطقی می‌یابد و سپس با  $R3$  جمع می‌شود و از آدرس حاصل، یک بایت خوانده شده و در  $R8$  ذخیره می‌شود:





LSL #3:  $R4 = 0x100$

$R3 + \text{Shifted } R4 = 0x4100$

Final Memory Address:  $0x4100$

$R8 = 0x000000FF$

c) نوع آدرس دهی در این بخش: Post-indexed addressing mode with fixed offset

در این حالت ابتدا محتوای آدرس حافظه‌ای که  $R3$  اشاره می‌کند خوانده می‌شود و در  $R7$  ذخیره می‌شود. سپس مقدار  $R4$  با  $R3$  جمع می‌شود و در  $R3$  ذخیره می‌شود:

Memory Address =  $0x4000$

$R7 = 0xFFFFFFFF$

$R3 = R3 + R4 = 0x4020$

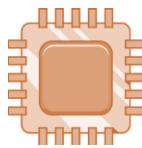
d) نوع آدرس دهی در این بخش: Scaled register post-indexed  
در این حالت ابتدا یک بایت از محتوای  $R6$  در خانه‌ای که رجیستر  $R3$  به آن اشاره می‌کند ذخیره می‌شود سپس مقدار  $R4$  دو بار به راست شیفت حسابی می‌یابد و سپس با  $R3$  جمع می‌شود و حاصل در  $R3$  ذخیره می‌شود:

Memory Address =  $0x4000$

Content of memory address =  $0x44$

$R4 \text{ ASR } \#2 = 0x08$

$R3 = R3 + (R4 \text{ ASR } \#2) = 0x4008$



## 6) 4 مورد از قوانین استاندارد AAPCS برای پیاده‌سازی توابع را نام ببرید

پاسخ:

- آرگومان‌های تابع باید از طریق رجیسترها R0 تا R3 فرستاده شوند.
- مقدار بازگشتی باید در R0 (و اگر مقدار 64 بیتی است) قرار گیرد.
- توابع می‌توانند از رجیسترها R4 تا R11 و R8 برای ذخیره اطلاعات موقت استفاده کنند. البته مقادیر این رجیسترها هنگام ورود باید ذخیره شود و قبل از بازگشت بازگردانی شوند.
- استک باید به صورت **full ascending** استفاده شود.



دانشکده مهندسی  
کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر



دانشگاه صنعتی امیرکبیر  
(پلی‌تکنیک تهران)

# دستورکار پروژه نهایی ریزپردازندۀ و زبان اسambilی (پاییز 1400)

طراحان:

مهردی قیاسی

ارشیا رحیمی

بهداد منصوری

سعیدمعین ایوبزاده

محمدرضا صادقیان

## توضیحات:

- در ادامه فایل، توضیحات 3 پروژه امتیازی متفاوت از بخش‌های مختلف درس آورده شده است و هر فرد تنها می‌تواند 1 پروژه را انتخاب کند و انجام دهد.  
(نمره امتیازی هر پروژه مشخص شده است)
- پیاده‌سازی پروژه‌ها باید به صورت فردی انجام شود. در صورت مشاهده تقلب، برای همه افراد نمره صفر لحاظ خواهد شد.
- همه پروژه‌ها شامل تحويل آنلاین هستند، به همین دلیل دلاین ارسال پروژه تا پایان روز پنج شنبه 7 ام بهمن ماه می‌باشد و به هیچ عنوان تمدید نخواهد شد.

## پروژه اول: X-O (2 نمره)

### شرح کلی پروژه:

به علت شیوع کرونا و قرنطینه اجباری حوصله غزل و یزدان سر رفته است. آنها تصمیم گرفته اند که با بازی کردن X-O (دوز خودمان) خود را سرگرم کنند. در این پروژه قصد داریم به کمک ماژول Xbee ، Keypad و برد آردوینو و چند LED (در حالت امتیازی LCD) بستری طراحی کنیم که غزل و یزدان با حفظ فاصله اجتماعی بتوانند با یکدیگر دوز بازی کنند.

### اهداف پروژه:

- یادگیری کارکردن با ماژول Xbee
- نوشتن کد در آردوینو
- اتصال دو یا چند (برای حالت امتیازی) برد آردوینو به یکدیگر
- اتصال آردوینو به Keypad و کدنویسی با استفاده از کتابخانه Keypad.h
- شبیه سازی اتصال دو پورت اتصال مجازی (COM) به یکدیگر برای شبیه سازی ماژول های کنترل از راه دور در پروتئوس (Proteus)

! Xbee یا Zigbee



زیگبی نوعی شبکه بی‌سیم است که در انسستیتو مهندسان برق و الکترونیک با کد IEEE 802.15.4 استانداردسازی شده است. در فناوری ZigBee طراحی به نحوی است که سیگنال‌های رادیویی دیجیتال با انرژی کم (low power) در شبکه‌های شخصی (PAN) با وسعت و برد کم توزیع شده و پهنای باند به دست

آمده نیز کم است. در قبال از دست دادن پهنانی باند انتقالی و برد پوشش، قیمت مقرر به صرفه و مصرف بسیار کم انرژی به دست می آید.

شبکه بی سیم ZigBee نسبت به شبکه های بی سیم دیگر نظری WiFi و Bluetooth ارزان تر است و در شبکه هایی کاربرد دارد که ارسال داده با نرخ و مصرف انرژی پایین مورد نیاز باشد. از طرفی زیگبی در مقایسه با واي فاي و بلوتوث سرعت انتقال داده کمتری دارد، علت اين تفاوت را می توان در هدف از طراحی اين فناوري دانست که به منظور صرف جوبي در مصرف انرژي بوده و برای ايجاد شبکه هایي مورد استفاده قرار می گيرد که به انتقال داده های کم، بهره وری انرژي و شبکه اى ايمن نياز دارند. به كمك اين نوع طراحی، استفاده از ZigBee، به نسبت دیگر انواع شبکه های بی سیم هزينه هی کمتری دارد.

به طور کلی بسیاری از افراد دو اصطلاح XBee و ZigBee اشتباه گرفته و به جای هم استفاده می کنند؛ همانطور که اشاره شد ZigBee پروتکل استاندارد برای شبکه های بی سیم است. در حالی که XBee محصولی است که از پروتکل های مختلف ارتباط بی سیم از جمله Wi-Fi، ZigBee و پشتيبانی می کند. در اين پروژه به طور عمده روی مازول XBee / XBee-PRO ZigBee تمرکز شده که از فريمور تشکيل شده است.



شکل 1 مازول XBee

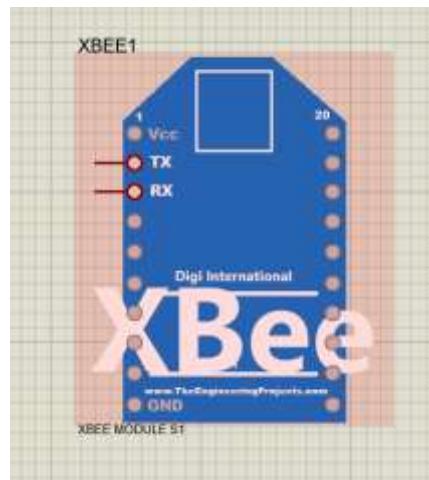
سوال: محدوده کاري (range) مازول ايکس بي معمولاً چقدر است؟ 

گام اول پروژه

كتابخانه مازول ايکس بي را از [لينك](#) دانلود کرده و آن را به نرم افرا Proteus اضافه کنيد.

(اگر از Proteus 8 Professional استفاده می کنید پوشه کتابخانه های آن در C:\Program Files (x86)\Labcenter Electronics\Proteus 8 Professional\DATA قرار دارد).

همانطور که مشاهده می شود ماژول شبیه سازی شده است و تنها در آن پورت های Rx و Tx قرار دارند.



شکل 2- شبیه سازی ماژول در Proteus

در گام اول دو XBee را به ترمینال مجازی (Virtual Terminal) متصل کنید ، پورت های ارتباطی آن ها را دو پورت مختلف (مثلا COM1 و COM2) قرار دهید ، با استفاده از نرم افزار Virtual Serial Port Driver در قسمت Merge ، دو پورت انتخاب شده را تلفیق کنید. حال ورودی در یکی از ترمینال ها وارد کنید، اتصال دو ماژول به یکدیگر را بررسی کنید.

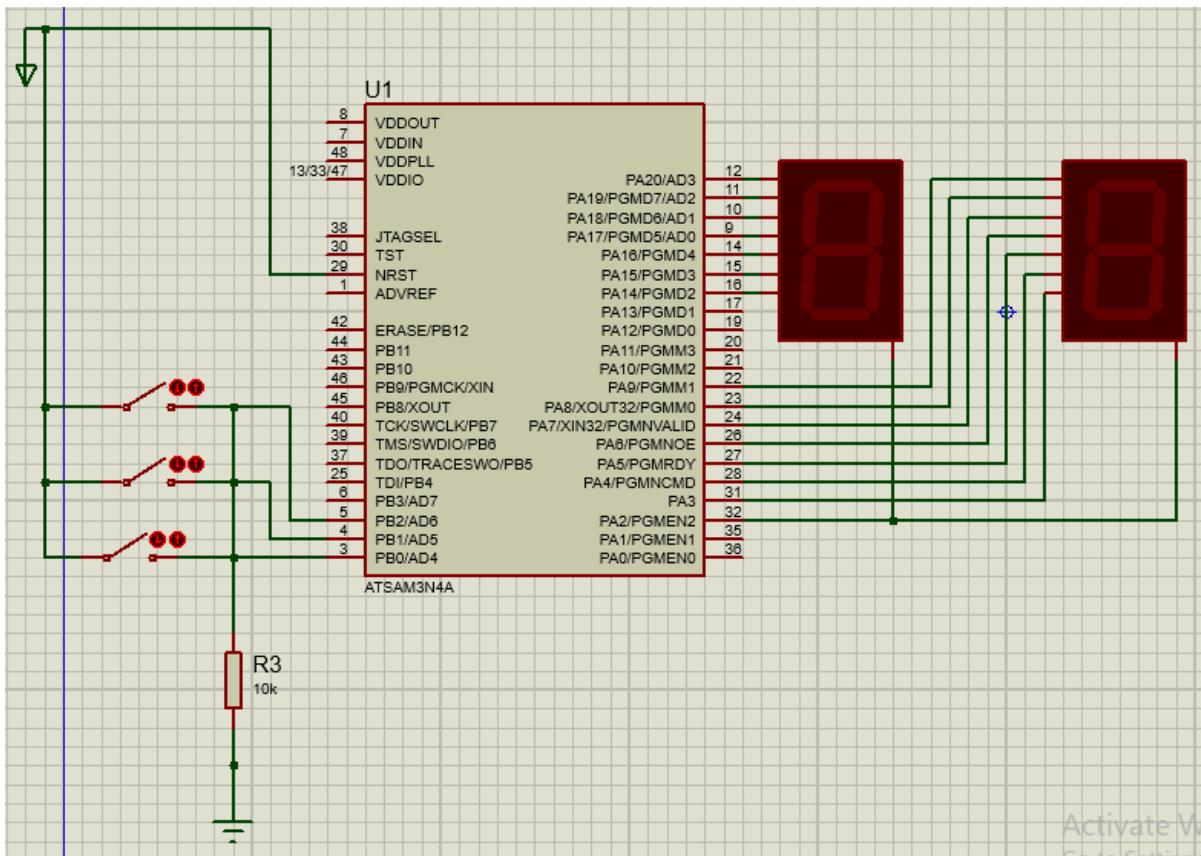
## گام دوم پروژه

حال که توانستیم دو ماژول ایکس بی را به هم متصل کنیم ، بنظر می رسد می توانیم مشکل غزل و یزدان را تا حد زیادی حل کنیم ! در این قسمت از پروژه باید با استفاده از دو آردوینو و ماژول های ایکس بی و کیپد های متصل به آنها و همچنین تعدادی LED که نشان دهنده برد بازی دوز است (برای هر کدام) آردوینو ها را به گونه ای برنامه نویسی کنید که غزل و یزدان بتوانند با یکدیگر دوز بازی کنند و برای سلامتی و طول عمر شما دعا کنند.

توجه: نحوه پیاده سازی برد بازی می تواند تعدادی LED دو رنگه، LCD و یا تعدادی LED معمولی باشد. به پیاده سازی خلاقانه نمره اضافه تعلق می گیرد.

## پروژه دوم: ثانیه شمار اسambilی (1 نمره)

شرح کلی پروژه:



در این پروژه شما باید مطابق شکل زیر یک ثانیه شمار را با استفاده از Seven-Segment و یک پردازنده ATSAM3N4A پیاده سازی کنید.

کار کرد این ثانیه شمار بدین صورت است که اگر کلید اول (از بالا) باز باشد، ثانیه شمار خاموش خواهد بود و صفحات نمایش گر چیزی را نشان نخواهد داد. در صورت بسته شدن کلید اول ثانیه شمار روشن شده و شروع به شمردن می کند.

هر زمان که کلید دوم بسته شود، ثانیه شمار متوقف می شود و در صورت باز شدن مجدد آن، ثانیه شمار از همان شماره قبلی که در آن قرار داشت شروع به ادامه شمارش می کند. چنانچه در حین شمردن ثانیه ها ثانیه شمار

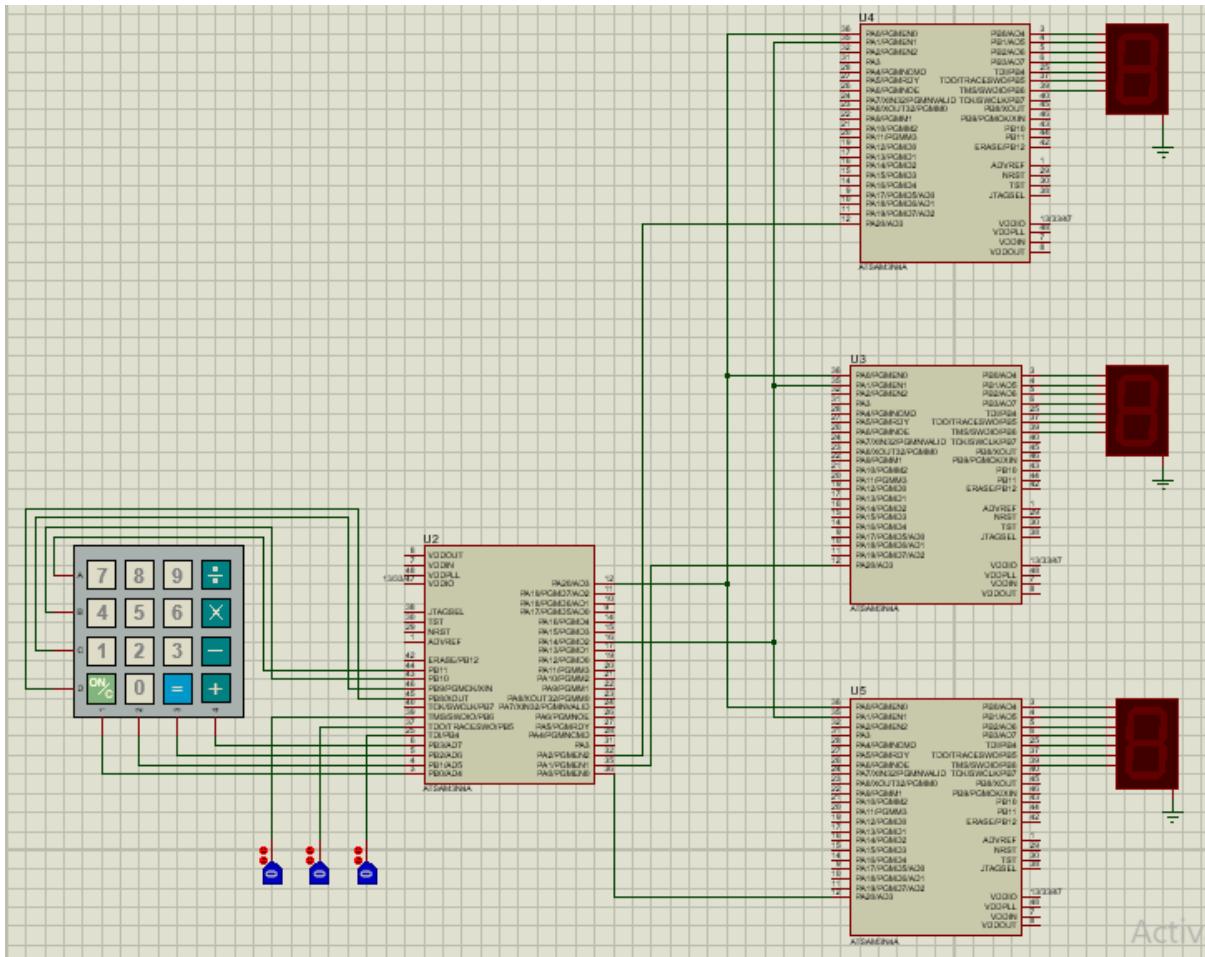
خاموش شود، پس از روشن شدن مجدد آن، باید از شماره ای که قبلا در آن قرار داشت شروع به شمارش کند.  
همچنین در صورت خاموش شدن پردازنده نیز ثانیه شمار باید قادر باشد شمارهای که قبل خاموش شدن  
پردازنده در آن بوده را بازیابی کند و به شمارش ادامه دهد.

کلید سوم برای **reset** کردن ثانیه شمار است و هرگاه بسته شود، ثانیه شمار صفرمی شود و تا زمانی که مجددا  
باز نشده، ثانیه شمار صفر باقی می‌ماند. چنانچه این کلید هرگز بسته نشود، ثانیه شمار تا مقدار 60 خواهد  
شمرد و سپس به طور خودکار به مقدار صفر بازمی‌گردد.

شما باید با برنامه ریزی کردن پردازنده به زبان **asmbl** قابلیت‌های شرح داده شده در بالا را پیاده سازی کنید.

## پروژه سوم: پیاده سازی پروتکل SPI (2 نمره)

شرح کلی پروژه:



اولین سیمی (ازبالا) که از سمت راست پردازنده master خارج می‌شود برای سیگنال کلک است. سیم دوم همان MOSI می‌باشد که به همه slave ها متصل می‌شود. و سه سیم پایینی در سمت راست پردازنده همان slave select ها هستند که active low می‌باشند. در این پروژه نیازی به پیاده سازی MISO نیست زیرا slave ها هیچگاه به master داده ارسال نمی‌کنند.

در پایین شکل، 3 عدد logic state قرار دارد که باید بتوانیم با آن ها slave مورد نظر را انتخاب کنیم. مثلاً اگر logic state سمت راست 1 و دو تای کناری صفر باشند، slave اول از بالا فعال است و slave های دیگر غیر فعال هستند. به دلیل اینکه slave ها هیچگاه داده‌ای برای master نمی‌فرستند، فعال بودن دو slave به صورت هم زمان امکان پذیر است.

نحوه کارکرد مدار به این صورت است که هرگاه در keypad سمت چپ عددی فشرده شد، پردازنده master آن را برای slave های فعال بفرستد و آن‌ها عدد را بر روی seven-segment ای روبروی آن‌ها قرار دارد نمایش دهند.

شما باید با برنامه ریزی پردازنده‌های master و slave با زبان اسمنبلی قابلیت گفته شده در بالا را پیاده سازی کنید.

**توجه:** نیازی نیست حتماً پایه‌های ورودی را دقیقاً مطابق شکل به دستگاه‌ها متصل کنید و می‌توانید از هر پایه ورودی/خروجی که خواستید استفاده کنید.

**توجه:** برای راحتی کار می‌توانید برای خواندن داده‌ها از پایه‌های PIO، از روش Polling استفاده کنید اما پیاده سازی پروژه‌ها به صورت interrupt driven امتیاز بیشتری خواهد داشت.

**موفق باشید**