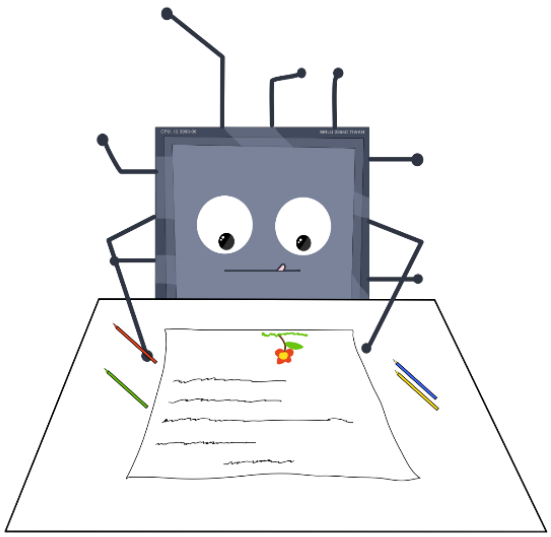
Department of Computer Engineering

Microprocessors and Assembly Language, Spring 2023, Dr. Farbeh

# Homework 1 – Solutions

Lec 1-6



## سوال ۱:

در ارتباط با بحث Microcontroller و Microprocessor به سوالات زیر پاسخ دهید:

الف) چند تا از برتری‌هایی که باعث شده‌است در سیستم‌های نهفته از Microcontroller استفاده شود را نام ببرید.

ب) تفاوت Microcontroller و Microprocessor چیست؟

پ) آیا استفاده از Microcontroller ها به جای Microprocessor ها بهینه‌تر است؟ توضیح دهید.

## پاسخ:

الف)

- تمامی حافظه‌ها و I/O ها در درون یک میکروکنترلر قرار دارد و نیازی نیست با این دیوایس‌ها را خریداری کرده و با هم اسمبل کنیم.
- با توجه به اینکه همه دیوایس‌ها را داراست، 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.
- مصرف کمتر انرژی

ب) در واقع microprocessor همان ic است که تنها cpu را شامل می‌شود و بقیه چیزهای سیستم(output,input,RAM,ROM و ...) از طریق یک bus به cpu متصل می‌گردند . قلب یک سیستم کامپیوتری است و حجم مداری سیستم بزرگتر است . به خاطر ارتباط‌های external انرژی بیشتری مصرف می‌کند . بر اساس معماری von neumann است. microcontroller با توجه به نیازهای سیستم های نهفته، با مصرف انرژی بسیار پایین و توان پردازشی کم ولی دسترسی به حافظه بالا طراحی می شود. توجه شود که این واحد چون نقش کنترلی نیز دارد، به آن microcontroller گفته می‌شود.

پ) در صورتی که نیاز به توان پردازشی کمی داشته باشیم، بله، بهینه‌تر است. به عنوان مثال از آن‌ها در سیستم‌های نهفته که نیاز به توان پردازشی کم و مصرف انرژی کمی دارد، استفاده می‌شود. اما اگر نیاز به قدرت پردازشی بالا و کار با حجم زیادی از داده‌ها داشته باشیم و توان مصرفی انرژی برایمان اهمیت کمتری داشته باشد، بهتر است که از microprocessor ها استفاده کنیم.

## سوال ۲:

درباره ISA به سوالات زیر پاسخ دهید:

الف) انواع دسته بندی ISA براساس نحوه خواندن Operand ها را بنویسید.

ب) انواع ISA را از نظر طول دستور نام برده و مزایا و معایب آن‌ها را بنویسید.

پ) پردازنده ما باید حداقل شامل کدام گروه از فانکشن‌ها باشد تا ISA کاملی به حساب آید؟

## پاسخ:

الف) براساس اینکه ISA عملوند (Operand)های خود را از کجا دریافت می‌کند به چهار دسته تقسیم می‌شوند:

- Stack : عملوندها از استک خوانده می‌شوند و نتیجه هم در همان ذخیره می‌شود. ( این سری پردازنده دیگر وجود خارجی ندارند.)
- Accumulator : رجیستر انباشتگر که همیشه یکی از ورودی‌های هر دستور و خروجی ALU در آن ذخیره می‌شود.
- Register-Memory : در این نوع معماری، عملوندها می‌توانند هم از رجیسترها و هم از حافظه اصلی (Memory) خوانده شوند. همچنین این معماری، از دستورات پیچیده تری پشتیبانی می‌کنند. یک نمونه کامپیوتر در این نوع معماری Intel x86 می‌باشد.
- Load-Store : برخلاف دسته سوم، این کامپیوترها خواندن و نوشتن از حافظه را فقط با دو دستور Load و Store انجام می‌دهند و عملوندها برای یک عملیات (مانند جمع) باید در رجیسترها ذخیره شده‌باشند و نمی‌توان در یک دستور هم از حافظه خواند و هم یک operation انجام شود.

ب)

**دستورات با طول متغیر:**

- معایب: به علت متغیر بودن طول دستور، کار fetch و decode کردن سخت‌تر می‌شود و این موضوع باعث پیچیدگی در خواندن دستورات می‌گردد.
- مزایا: می‌توان طول دستورها را متناسب با نیاز تغییر داد که این باعث انعطاف پذیری بیشتر می‌شود.

**دستورات با طول ثابت:**

- مزایا: به علت ثابت بودن طول دستور، کار fetch و decode کردن و همچنین pipelining آسان‌تر می‌شود.

پ) برای اینکه یک ISA کامل باشد باید دارای حداقل این سه گروه از فانکشن ها باشد:

- load / Store
- Control
- Arithmetic / Logic

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

## سوال ۳:

الف) به چه دلیل در پردازنده SAM3X8E به هنگام مد sleep فقط clock هسته متوقف می‌شود؟
ب) تفاوت مد Glitch filters و مد Debouncing در مدهای I/O را شرح دهید.
پ) سه مدل مختلف تایمر در این ریزپردازنده را نام ببرید و موارد استفاده از هر کدام را شرح دهید.

## پاسخ:

الف) به این علت که ممکن است حالتی مثل انتقال حجم انبوهی از داده وجود داشته‌باشد که فقط در ابتدای کار نیاز به هماهنگی هسته باشد و دیگر نیازی به هسته نباشد، پس برای جلوگیری از مشغول کردن پردازنده و مصرف انرژی، کلاک هسته را متوقف می‌کنیم.

ب) در مد Glitch filters ، با فیلتر کردن پالس‌های ناخواسته یا همان نویزها، پالس‌های ورودی را به داده صفر و یک تبدیل می‌کنیم، اما مد Debouncing به عنوان مثال در هنگام استفاده از دکمه‌ها به کار می‌رود و با فیلتر کردن تغییرات در بازه مشخصی که سیگنال ورودی به ثبات برسد، سیگنال درست را دریافت می‌کند.

پ)

Real_time timer(RTT)

- یک شمارنده ۳۲ بیتی که ثانیه‌های سپری‌شده را می‌شمارد(زمان واقعی را به ما نشان می‌دهد و هر ثانیه یکی به آن اضافه می‌شود.). می‌توان از این timer به جای timer دست‌ساز خودمان در micro برای آن‌که زمان واقعی را اندازه بگیریم، استفاده کنیم. به صورت periodic، interrupt می‌دهد( مثلا هر روز، هر ماه و ...) . reg به نام RTT_VR وجود دارد که value فعلی را می‌توان از آن خواند. این register، read_only بوده و هر بار که سیستم روشن شود، مقدار قبلی از آن پاک شده و از صفر شروع به شمارش می‌کند.

Real_time clock(RTC)

- با مصرف انرژی بسیار کمی ساعت می‌سازد. در حالت RTT فقط ثانیه شمرده می‌شود ولی در این حالت هم ثانیه، هم دقیقه، هم ساعت و هم روز را حساب می‌کند. در واقع یک تقویم میلادی نیز دارد.هر وقت این system، reset شود و بعد دوباره روشن کنیم، به مقدار default خود برمی‌گردد و تا حدود ۲۰۰ سال میلادی را محاسبه می‌کند. در این حالت، به صورت periodic، interrupt داریم. (مثلا هر هفته برای ما آلارم بگذارد.)

Watchdog timer(WDT)

- هنگامی‌که سیستم قفل شده(در حالت deadlock است)، این timer باعث می‌شود از این مخمصه بیرون بیاید. به این صورت که ما مقدار حدودی که یک instruction نیاز دارد را در این تایمر ثبت می‌کنیم. اگر به هر دلیلی این زمان سپری شد ولی همچنان برنامه ما تمام نشده بود، مثلا اگر اروری رخ بدهد و برنامه نتواند جلو برود، این WDT باعث می‌شود برنامه reset شده و در نتیجه از این توقف بیرون بیاییم( این کار را با وقفه انجام می‌دهد). باید حواسمان باشد وقتی کارمان تمام شد، این تایمر را غیرفعال کنیم. WDT یک شمارنده ۱۲ بیتی است.

## سوال ۴:

سوالات زیر مربوط به وقفه‌ها را پاسخ دهید:

الف) در NVIC وقفه‌ها با توجه به نوع priority شان، در چهار دسته می‌توانند باشند. آن‌ها را نام برده و مختصری توضیح دهید.

ب) چگونه می‌توان تعداد بیت‌های اختصاص داده‌شده برای اولویت‌بندی وقفه‌ها را محاسبه کرد؟

## پاسخ:

الف) در NVIC وقفه‌ها را از روی priority یا اولویت می‌توان به ۴ دسته تقسیم کرد. پیش از توضیح می‌دانیم که هر چه عدد priority کوچک‌تر باشد اولویت آن بیشتر است. اولین دستور reset است که اولویتی برابر با ۳- دارد که بیشترین اولویت است و به این معنی است که دستگاه در حال انجام هر کاری که باشد اگر این دکمه فشرده شود، آن را رها کرده و reset می‌شود. بعدی Non-maskable Interrupt بوده که همانطور که از اسم آن مشخص است، نمی‌توان آن را در نظر نگرفت و یا برای انجام یک وقفه دیگر آن را متوقف کرد و دارای اولویت ۲- است. بعدی Hard fault است که اولویت ۱- دارد و این ۳ نوع را نمی‌توان دست‌کاری کرد و تغییر داد. دسته آخر Configurableها هستند که اولویت‌های آن‌ها قابل تنظیم و دستکاری است و می‌توان به آن‌ها اعداد بی-علامت را نسبت داد.

ب) می‌توانیم در value مربوط به اولویت وقفه مورد نظر، عدد 0xFF را بنویسیم( ۸ بیت ۱ = ۲۵۵) و سپس آن را دوباره بخوانیم. اگر از نظر سخت افزاری ۸ بیت پیاده سازی شده باشد، دوباره همان ۲۵۵ خوانده می‌شود. اما اگر بعد از خواندن، مثلا عدد 0xE0 را ببینیم، به این معناست که این وقفه فقط ۳ بیت برای اولویت دارد؛ چون ۵ بیت کم ارزش را صفر در نظر گرفته‌است و در واقع این بیت‌ها نادیده گرفته شده‌اند.

## سوال ۵:

فرض کنید سیستم ما دو نوع وقفه A و B دارد که وقفه نوع A اولویت بالاتری دارد و برای انجام ISR آن به ۴۵ کلاک نیاز داریم در حالی که وقفه B به ۳۰ کلاک نیاز دارد. حال فرض کنید وقفه B رخ می‌دهد و در حین انجام آن A رخ می‌دهد. با فرض اینکه برای load کردن رجیسترها از حافظه به ۱۵ کلاک و برای save کردن رجیستر ها به ۱۵ کلاک نیاز داشته باشیم ، بازده CPU ما از زمانی که کنترل را از برنامه اصلی می‌گیرد تا بازگشت به آن چقدر است ؟

## پاسخ:

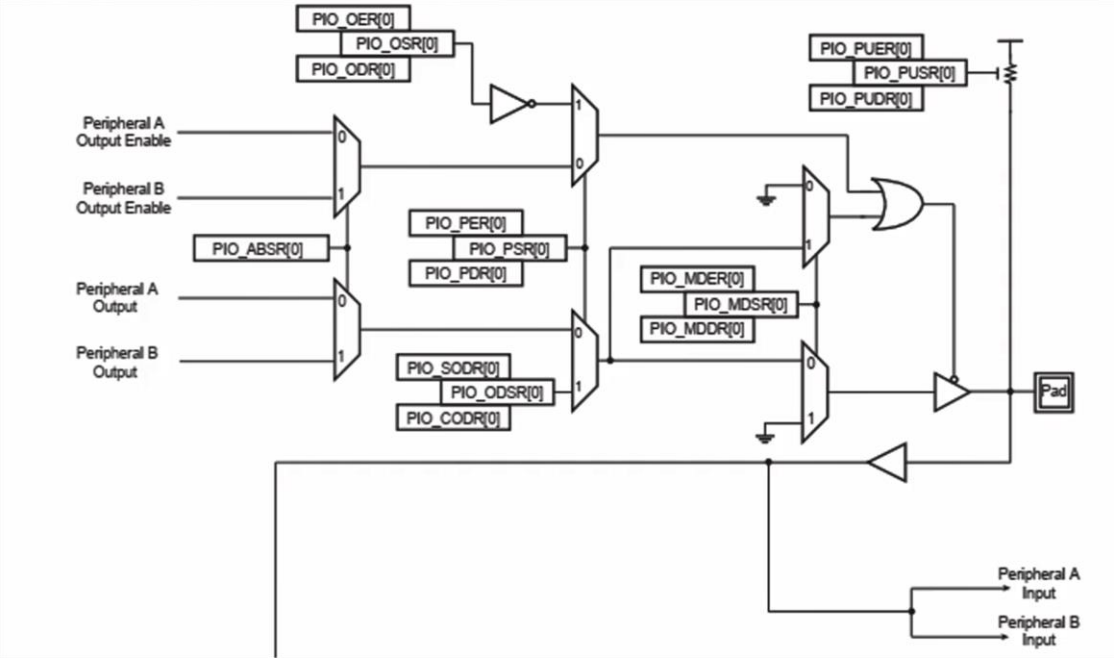
در ابتدا باید ۱۵ ثانیه رجیسترها سیو شوند و بعد در حین انجام B وقفه A رخ می‌دهد؛ پس باز باید رجیسترهای وقفه B ذخیره شود و بعد از آن وقفه A اجرا می‌شود و سپس رجیسترهای وقفه B لود می‌شود و سپس ISR آن اجرا می‌شود و بعد از آن رجیستر های برنامه اصلی لود می‌شود. در تمامی این مراحل، ما صرفا دو ISR اجرا کردیم که در مجموع ۷۵ کلاک مفید به حساب می‌آید.

$$15 + 15 + 45 + 15 + 30 + 15 \quad = 135$$

در نتیجه بازده CPU نزدیک ۰.۵۵ بدست می آید.

## سوال ۶:

توجه به موارد ذکر شده و شکل زیر به سوالات پاسخ دهید. (در هر مورد، فقط بیت صفرم رجیسترها را تعیین کنید.)




الف) در صورتی که PIO_MDSR = 1, PIO_PUSR = 0 باشد، خروجی نهایی مدار قسمت بالا چه خواهد بود؟ توضیح دهید .

ب) وضعیت PIO_PSR, PIO_OSR, PIO_MDSR را برای این‌که بخواهیم داده‌ای را از PIO_ODSR به خروجی منتقل کنیم، تعیین کنید.

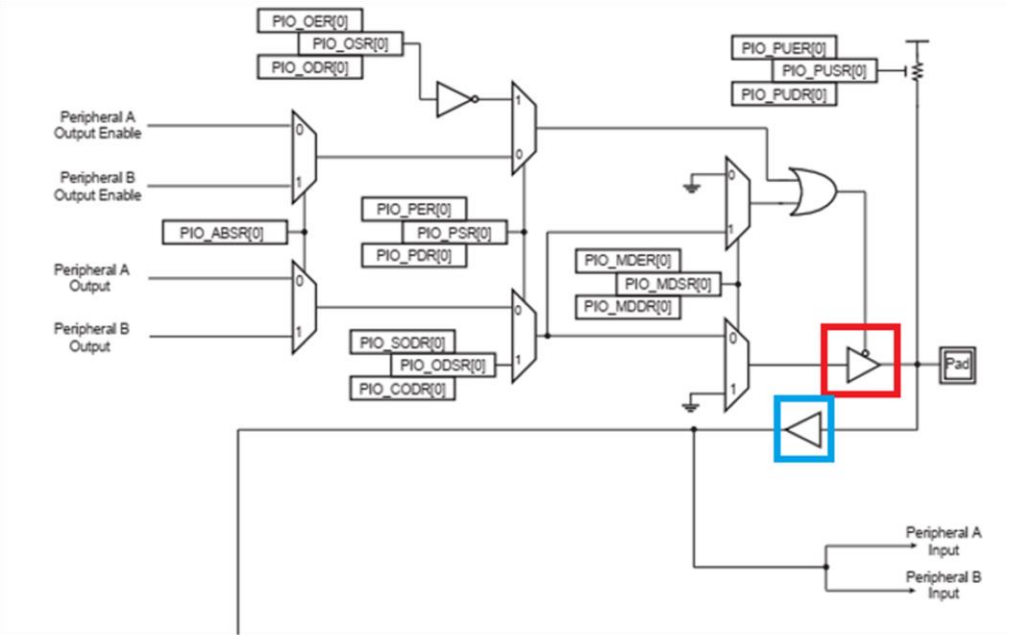
پاسخ:

الف) در صورتی که PIO_PUSR = 1 باشد، مدار در حالت multi-drive قرار می‌گیرد و ورودی بافر سه حالته، بیت 0 می‌باشد. حالا با توجه به این که پایه enable بافر چه مقداری داشته باشد، خروجی تفاوت می‌کند. در صورتی که ورودی پایه enable صفر باشد، بافر فعال می‌شود و خروجی صفر خواهد بود، اما در صورتی که ورودی پایه enable یک باشد، خروجی high-z خواهد بود.

ب) برای اینکه بخواهیم داده ای را به خروجی منتقل کنیم، باید PIO_OSR = 1 باشد تا بعد از عبور از گیت not، به صورت بیت صفر به گیت OR وارد شود. همچنین PIO_MDSR = 0 باشد تا هم حالت multi-drive نباشد و هم ورودی دیگر گیت OR، صفر شود تا بافر را فعال کند. PIO_PSR هم باید 1 باشد.

## سوال ۷(امتیازی):

با توجه به تصویر روبه رو که منطق کنترلی پایه شماره صفر واحد PIO را نشان می‌دهد، به سوالات زیر پاسخ دهید.




الف) چرا بافر مشخص‌شده با رنگ قرمز سه حالته و بافر مشخص شده با رنگ آبی دو حالته است؟

ب) اگر بخواهیم از طریق مدار بالا داده‌ای را از Peripheral A به Peripheral B ارسال کنیم، مقادیر بیت اول در تمام رجیسترهای Status را برای انجام این‌کار تعیین کنید.(pull-up و multi-drive غیرفعال است.)

پاسخ:

الف) به دلیل این‌که بتوانیم از پین به عنوان ورودی مقدار بخوانیم باید بافر بالایی سه حالته باشد تا خروجی بافر بتواند HI-Z بشود. به عبارتی از نیمه بالا، سیگنالی وارد پین نشود. از طرفی بافر پایینی دو حالته است به این دلیل که ممکن است بخواهیم از خروجی مدار بالا در مدار پایین استفاده کنیم.(زمانی که از peripheralهای داخلی میکرو برای اهداف داخلی خودمان استفاده کنیم.)

ب) PIO_PUSR = 0, PIO_MDSR = 0, PIO_ODSR = 0/1, PIO_PSR = 0, PIO_OSR = 1, PIO_ABSR = 0