فربد فولادی-98243045

عرفان رفیعی اسکوئی-98243027

سوال اول)

مجموعه دستورالعمل A64 توسط معماری Armv8-A پشتیبانی می شود. ویژگی های کلیدی A64 عبارتند از:

1. یک decode table خوب بر اساس یک مشخص کننده 5 بیتی.
2. Semantic دستورالعمل ها به طور گسترده شبیه به A32 و T32 است.
3. 31 رجیستر 64 بیتی همه منظوره که همیشه در دسترس هستند.
4. شمارنده برنامه و اشاره گر پشته، رجیسترهای عمومی نیستند.

دستورالعمل‌های A32 که در معماری‌های پیش از Armv8 به عنوان دستورالعمل‌های Arm شناخته می‌شوند، 32 بیت عرض دارند و بر روی مرزهای 4 بایتی تراز شده‌اند. دستورالعمل های A32 توسط هر دو معماری A-profile و R-profile پشتیبانی می شوند.

A32 به طور سنتی در برنامه هایی که به بالاترین عملکرد نیاز دارند یا برای رسیدگی به استثناهای سخت افزاری مانند وقفه ها و راه اندازی پردازنده استفاده می شد. با معرفی فناوری Thumb-2 بسیاری از قابلیت های آن در T32 قرار گرفت.

بیشتر دستورالعمل های A32 فقط زمانی اجرا می شوند که دستورالعمل های قبلی یک کد شرط خاص را تنظیم کرده باشند. این بدان معنی است که دستورالعمل‌ها تنها در صورتی تأثیر عادی خود را بر عملکرد مدل برنامه‌نویس، حافظه و پردازنده‌های مشترک دارند که پرچم‌های N، Z، C و V شرایط مشخص‌شده در دستورالعمل را برآورده کنند. اگر پرچم ها این شرط را برآورده نکنند، دستورالعمل به عنوان یک NOP عمل می کند. این بدان معنی است که اجرا به طور عادی به دستورالعمل بعدی پیش می رود، از جمله بررسی های مربوطه برای استثناهایی که انجام می شود، اما هیچ اثر دیگری ندارد. این اجرای مشروط دستورالعمل ها اجازه می دهد تا بخش های کوچکی از دستورات if و while بدون استفاده از دستورالعمل های شاخه کدگذاری شوند.

دسته ی بعدی T32 نام دارد که خودش دو instruction متفاوت را داراس یکی Thumb-1 و دیگری Thumb-2 :

Thumb : (معروف به Thumb-1) 16 رجیستر را به پایین (R0-R7) و بالاتر (R8-R12، SP، LR، PC) تقسیم می کند، اکثر دستورالعمل ها فقط می توانند به مجموعه پایین دسترسی داشته باشند، در حالی که فقط برخی می توانند به مجموعه بالاتر دسترسی داشته باشند. فقط اپکدهای 2 بایتی در دستگاه‌های ارزان‌قیمتی که دارای گذرگاه ۱۶ بیتی هستند (و باید دسترسی به کلمه ۳۲ بیتی را در دو مرحله انجام دهند) زمانی که کدهای عملیاتی ۲ بایتی را اجرا می‌کنند، بهتر عمل می‌کنند. Thumb را می توان به عنوان یک اصطلاح خانوادگی برای هر دو Thumb-1 همراه با Thumb-2 استفاده کرد، یا گاهی اوقات Thumb را می توان فقط برای Thumb-1 استفاده کرد. Thumb-1 یک اصطلاح رسمی Arm نیست، فقط چیزی است که توسط مردم برای ایجاد تمایز بین خانواده Thumb هر دو ISA و اولین ISA Thumb استفاده می‌شود. دستورالعمل ها در ARM می توانند پسوند s اختیاری برای به روز رسانی رجیستر CPSR داشته باشند (به عنوان مثال

دستورالعمل ands، orrs، movs، adds، subs)، در حالی که در Thumb-1 s همیشه روشن است و رجیستر CPSR را همیشه ذخیره می کند.

Thumb-2 : یک پسوند Thumb است و می تواند مانند ARM به همه رجیسترها دسترسی داشته باشد، دارای کدهای عملیاتی 4 بایتی با کمی تفاوت در مقایسه با ARM است. در اسمبلی، Opcode باریک 2 بایتی Thumb-1 و اپکد گسترده 4 بایتی Thumb-2 را می توان با پسوند .n و .w (مثال orr.w) تعیین کرد. فرمت‌ها/رمزگذاری‌های اپکد ARM و Thumb-2 متفاوت هستند و قابلیت‌های آنها نیز متفاوت است. تفاوت دیگر دستورالعمل های پردازش داده است. هر دو ARM و Thumb-2 از 8 بیت فوری پشتیبانی می کنند، در حالی که ARM می تواند بیت ها را فقط به سمت راست و فقط با بیت های زوج بچرخاند، در حالی که Thumb می تواند چرخش به چپ و تعداد بیت های زوج/فرد انجام دهد و در بالای آن، الگوهای بایت تکرار شونده امکان پذیر است.

منبع:

https://developer.arm.com/architectures/instruction-sets#:~:text=Arm%20Instruction%20Set%20Architecture,code%20density%20for%20user%20code

سوال دوم)

ابتدا به تعریف زبان ماشین می پردازیم:

در زبان ماشین یا کد ماشین، دستورالعمل ها در الگوهای باینری نوشته می شوند، یعنی ترکیبی از ارقام باینری 1 و 0، که به عنوان سطوح ولتاژ بالا و پایین ذخیره می شوند. این پایین ترین سطح زبان های برنامه نویسی است و زبانی است که یک میکروکنترلر یا ریزپردازنده واقعاً آن را می فهمد.

حال به تعریف زبان اسمبلی و زبان های سطح بالا می پردازیم و در ادامه بررسی می کنیم که کدام یک برای میکروکنترلرها بهتر است:

زبان اسمبلی:

سطح بعدی زبان برنامه نویسی، زبان اسمبلی است. از آنجایی که زبان ماشین یا کد شامل تمام دستورالعمل‌های 1 و 0 است، برنامه‌نویسی با استفاده از آن برای انسان‌ها بسیار دشوار است.

زبان اسمبلی یک نمایش شبه انگلیسی از زبان ماشین است. زبان اسمبلی میکروکنترلر ها ترکیبی از کلماتی شبیه به انگلیسی به نام‌های Mnemonics و کدهای هگزادسیمال است.

همچنین یک زبان سطح پایین است و نیاز به درک گسترده ای از معماری میکروکنترلر ها دارد.

زبان های سطح بالا:

نام زبان سطح بالا به این معنی است که ما نیازی به نگرانی در مورد معماری یا سایر جزئیات داخلی یک میکروکنترلر نداریم و آنها از کلمات و عباراتی استفاده می کنند که به راحتی برای انسان قابل درک است.

چند نمونه از زبان های سطح بالا عبارتند از BASIC, C , Pascal, C++, و Java. حال برای این که این زبان ها به زبان ماشین تبدیل شوند برنامه ای به نام کامپایلر، برنامه های نوشته شده به زبان های سطح بالا را به کد ماشین تبدیل می کند.

حال به این می پردازیم که چرا بهتر است از زبان اسمبلی استفاده کنیم تا یک زبان سطح بالا با این که این زبان ها ساده ترند:

1. برنامه های نوشته شده در اسمبلی سریعتر اجرا می شوند و حافظه کمتری را اشغال می کنند.
2. با کمک زبان اسمبلی می توانیم مستقیماً از تمام ویژگی های یک میکروکنترلر بهره برداری کنیم.
3. با استفاده از زبان اسمبلی می توانیم کنترل مستقیم و دقیقی بر تمامی منابع میکروکنترلر مانند پورت های ورودی/خروجی، RAM و غیره داشته باشیم.
4. در مقایسه با زبان های سطح بالا، زبان اسمبلی قوانین و محدودیت های کمتری دارد.