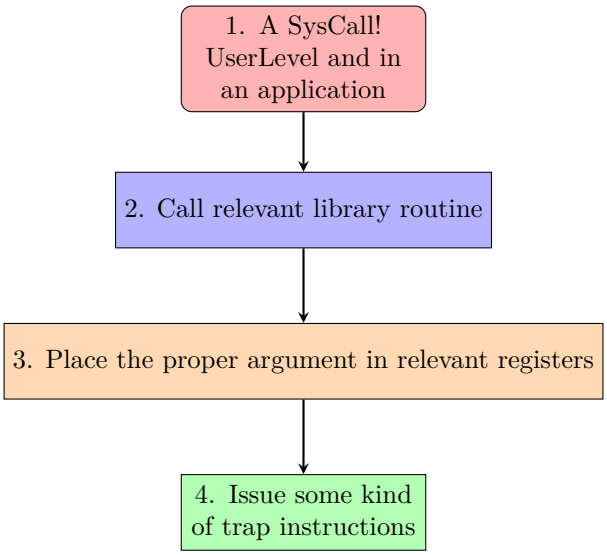
تشریح روند اتفاقاتی که در حین اجرای یک SysCall رخ میدهد

۷ آبان ۱۳۹۹

هدف استفاده از سیستم کال

وظیفه ی سیستم کال ایجاد یک انتقال حفاظت شده میان یک Application در User-mode به سیستم عامل در Kernel-mode است.
در واقع سیستم کال به سیستم عامل این دسترسی را میدهد تا کد سمت کاربر را بصورت حفاظت شده ای اجرا کند.
همچنین با پرش به کرنل و دستیابی به Privilege Level سیستم عامل برخی عملیات مشخصی را انجام دهد.

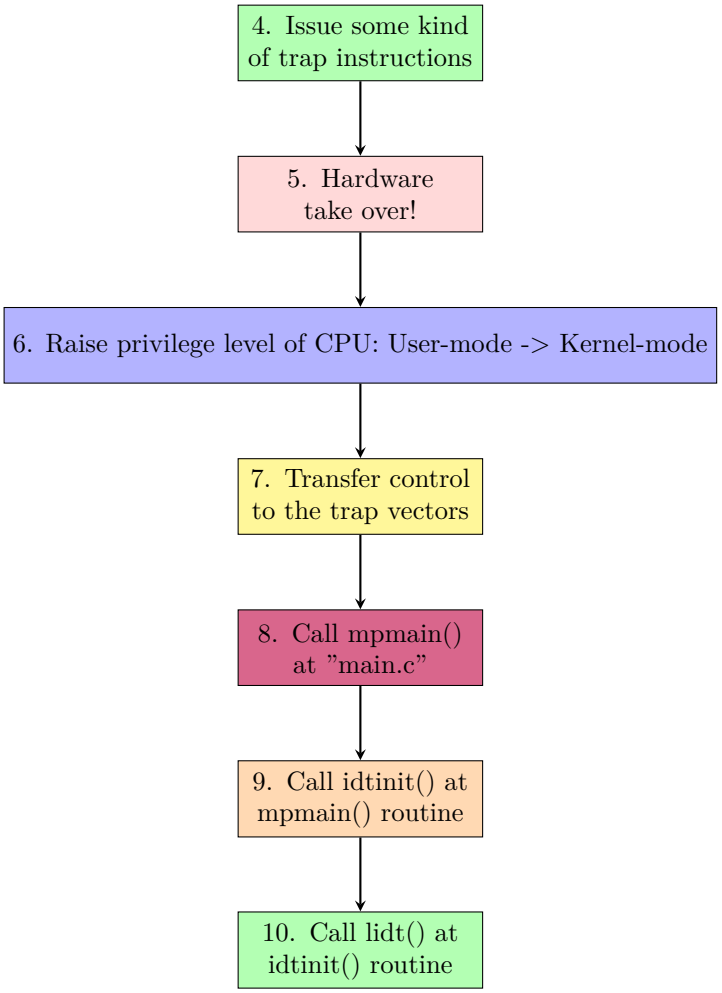
حال وارد جزئیات شده و نحوه ی پیاده سازی فراخوانی سیستمی را شرح می دهیم:



۱

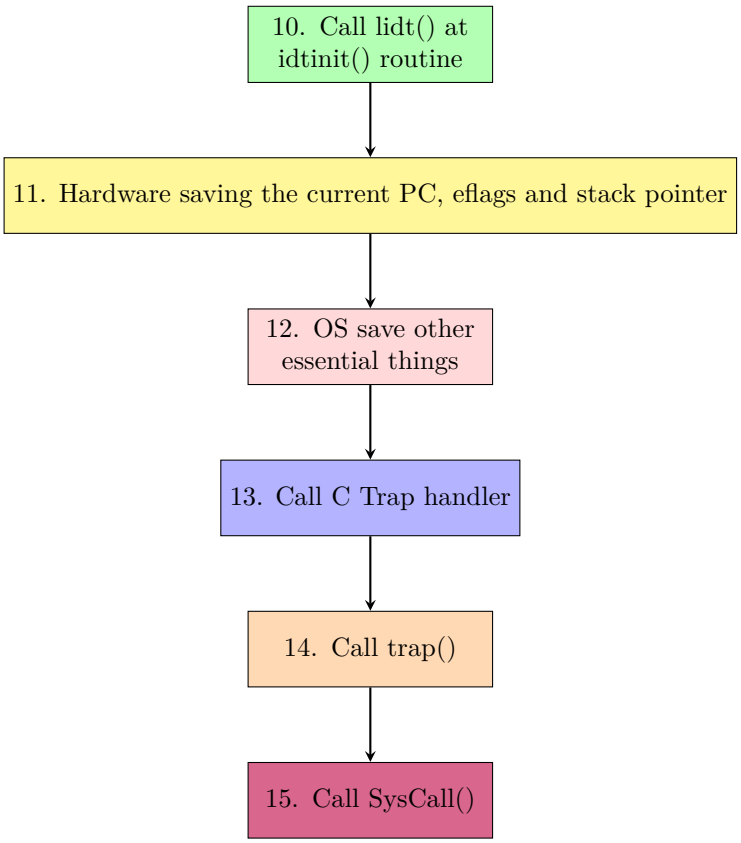
برای شفافتر شدن آنچه در فلوچارت اول آمد به توضیحات زیر توجه فرمائید:
در مرحله ی ۳ مقداری به رجیستر %eax وارد میشود که بعدتر توسط کرنل برای تشخیص اینکه چه سیستم کالی فراخوانی شده است استفاده میشود.
همچنین براساس آرگومانی که به دستور int داده میشود میتوان نوع trap را مشخص نمود.
درون سیستم عامل xv6 از 64 trap برای هندل کردن سیستم کال استفاده میشود. دیگر آرگومانهایی که به سیستم کال پاس داده میشوند، درون استک نگه داری میشوند.

با اجرای دستور int سخت افزار وارد عمل شده و تسکهای زیر را انجام میدهد:



۲

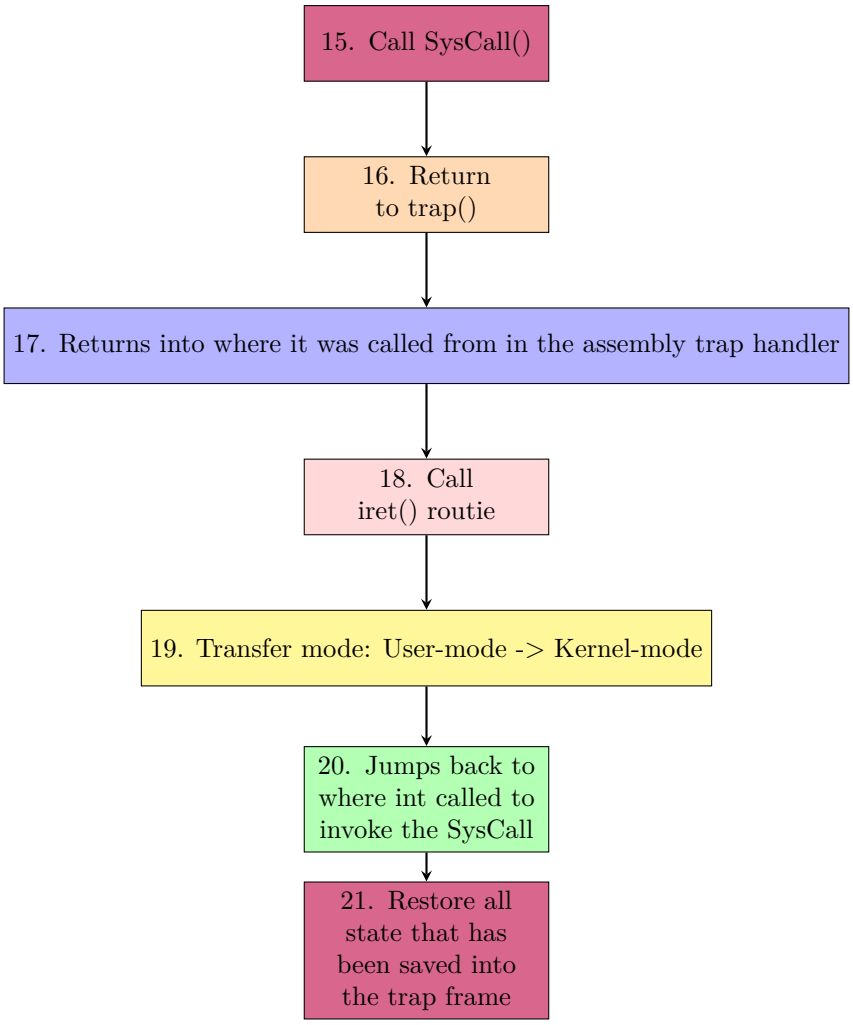
در مرحله ی هفتم کنترل به trap vectors منتقل می شود. لازم بذکر است که این وکتور در زمان boot شدن سیستم درون main.c توسط روتین idtinit() مقداردهی میشود.
درون روتین idtinit() از ماکروی SETGATE() استفاده شده که از آن برای ست کردن آرایه ی idt استفاده میشود.
آرایه ی idt(interrupt descriptor table) به محل کد مناسبی که به هنگام هر trap خاص نیاز است که اجرا شود اشاره میکند.
در اینجا سخت افزار شروع به ذخیره ی رجیسترها، پوینترها و متغیرهای حساس میکند و ادامه ی فرآیند را بوسیله ی فلوچارت زیر دنبال میکنیم:



۳

در مرحله ی دوازدهم سیستم عامل شروع به ذخیره سازی متغیرهای دیگر میکند. سیستم عامل با ران کردن vector64()، trap number را درون push stack میکند و بعد از آن alltraps() فراخوانی میشود که وظیفه ی ذخیره کردن بقیه ی موارد در trap frame را دارد.
C trap handler با فراخوانی trap() یک پوینتر به trap frame ارسال میکند و از میان آنها با چک کردن فیلد trap number از trap frame مشخص میکند که کدامیک اجرا شود و SysCall() را فرخوانی میکند.

توجه داشته باشید که در هریک از این مراحل چک میشود که اگر پروسه جاری kill شده، خارج شود و پروسه را پاک کرده و ادامه عملیات را انجام ندهد.
حال با استفاده از System call number که در رجیستر %eax ذخیره شده، شماره مناسب را از trap frame استخراج میکنیم و برای فراخوانی روتین مناسب که در جدول System call SysCall() آمده است از آن استفاده میکنیم.
نهایتا بررسی میشود که SysCall number در محدوده مجاز باشد و بعد با فراخوانی روتین مناسب SysCall هندل میشود.



٤

بعد از اجرای سیستم کال، برای مشخص کردن مسیر برگشت SysCall() یک مقدار صحیح برمیگرداند و آنرا در رجیستر %eax متعلق به trap frame قرار میدهد سپس به trap() بازگشته و نهایتا به سادگی به محلی که توسط assembly trap handler فراخونده شده باز میگردد.

با تشکر از توجه شما