TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI
KHOA ĐIỆN TỬ

Đề cương bài giảng

VI ĐIỀU KHIỂN

(Tài liệu lưu hành nội bộ)

Hà Nội - 08/2020

MỤC LỤC

CHƯƠNG 1 TỔNG QUAN VI ĐIỀU KHIỂN ARM

Chương này cung cấp các thông tin tổng quan về vi điều khiển lõi ARM nói chung và họ vi điều khiển STM32 nói riêng. Sau khi học xong, người học được trang bị các thông tin cơ bản về lịch sử phát triển của vi điều khiển ARM, các ứng dụng phổ biến của vi điều khiển ARM cũng như kiến trúc phần cứng của lõi ARM 32 bit.

1.1 Giới thiệu lịch sử phát triển của vi điều khiển ARM

Các thế hệ vi điều khiển ngày càng phát triển không ngừng nhằm đáp ứng các yêu cầu điều khiển, xử lý dữ liệu ngày càng lớn. Các vi điều khiển 8 bit rất phổ biến trong các ứng dụng điều khiển trong công nghiệp cũng như các sản phẩm dân dụng, các vi điều khiển 16 bit với khả năng đáp ứng cao hơn so với dòng vi điều khiển 8 bit, tuy nhiên với các yêu cầu điều khiển, khối lượng dữ liệu xử lý như hình ảnh trong các thiết bị điều khiển sinh học, các thiết bị giải trí như máy chụp ảnh kỹ thuật số, máy tính bảng, máy định vị dẫn đường, … thì các vi điều khiển 8 bit và 16 bit sẽ không đáp ứng được do không đủ không gian bộ nhớ để chứa dữ liệu, không đủ nhanh để xử lý dữ liệu,… để đáp ứng được các yêu cầu đó thì các thế hệ vi điều khiển 32 bit đã ra đời, trong đó dòng vi điều khiển phổ biến nhất hiện nay dựa trên lõi vi xử lý ARM.

Vi xử lý ARM là thành phần chủ lực làm nên thành công lớn của các hệ thống nhúng 32-bit. Các vi xử lý ARM được ứng dụng rộng rãi trong các điện thoại di động, máy tính bảng và các thiết bị di động khác. Phần lớn vi xử lý ARM có kiến trúc RISC (Reduced Instruction Set Computing), cho phép tiêu hao năng lượng thấp nên là một lựa chọn lý tưởng cho các hệ thống nhúng.

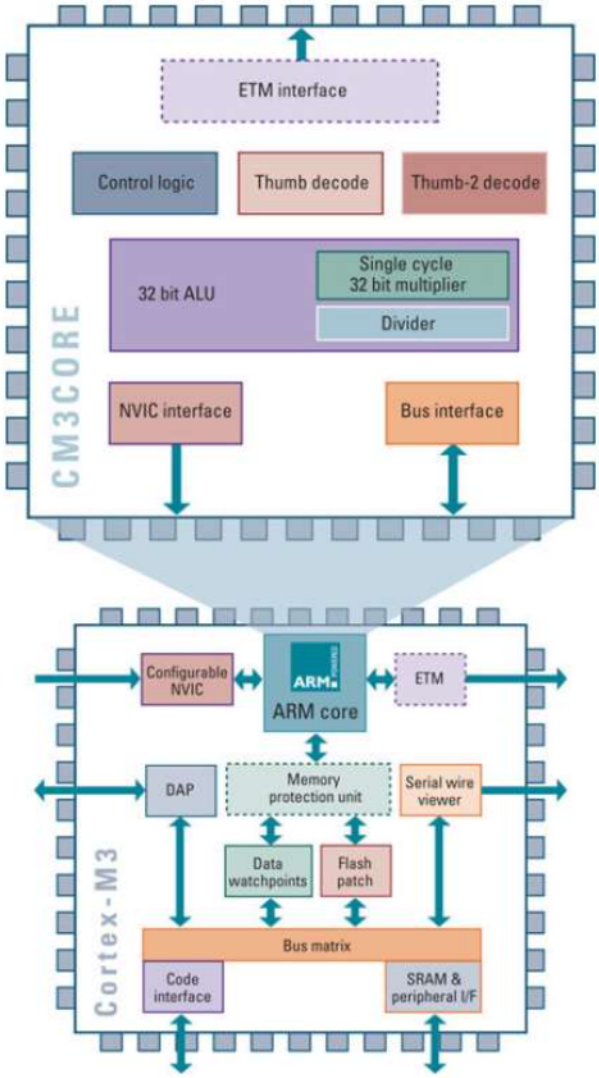
1.2 Kiến trúc phần cứng của vi điều khiển ARM 32 bit

1.2.1 Giới thiệu vi điều khiển ARM 32 bit

Kiến trúc ARM là một tập hợp các thông số kỹ thuật liên quan đến tập lệnh, mô hình thực thi, tổ chức và bố trí bộ nhớ, các chu trình lệnh và hơn thế nữa, mô tả chính xác một máy sẽ triển khai kiến trúc nói trên. Nếu trình biên dịch của bạn có thể tạo lắp ráp hướng dẫn cho kiến trúc đó, nó có thể tạo mã máy cho tất cả các máy thực tế đó (hay còn gọi là bộ xử lý) triển khai kiến trúc đã cho đó. Cortex-M là một họ lõi vật lý được thiết kế để tích hợp thêm với silicon dành riêng cho nhà cung cấp các thiết bị để tạo thành một bộ vi điều khiển hoàn thiện. Cách thức hoạt động của một lõi không chỉ được xác định bởi sự liên quan của nó Kiến trúc ARM (ví dụ: ARMv7-M), mà còn bởi các thiết bị ngoại vi và khả năng phần cứng tích hợp được xác định bởi nhà sản xuất silicon.

Ví dụ: kiến trúc lõi Cortex-M4 được thiết kế để hỗ trợ các hoạt động truy cập dữ liệu bit trong hai vùng bộ nhớ cụ thể bằng cách sử dụng một tính năng được gọi là dải bit, nhưng việc bổ sung tính năng đó hay không còn tùy thuộc vào việc triển khai thực

tế. STM32F4 là một họ MCU dựa trên lõi Cortex-M4 triển khai tính năng dải bit này. Hình 1 cho thấy rõ ràng mối quan hệ giữa MCU dựa trên Cortex-M3 và lõi Cortex-M3 của nó.



Hình 1.1. Mối quan hệ giữa lõi Cortex-M3 và bộ vi điều khiển dựa trên lõi Cortex-M3

ARM Holdings là một công ty của Anh[1] phát triển tập hướng dẫn và kiến trúc cho các sản phẩm dựa trên ARM nhưng không sản xuất thiết bị. Đây là một khía cạnh thực sự quan trọng của ARM thế giới và lý do tại sao có nhiều nhà sản xuất silicon phát triển, sản xuất và bán vi điều khiển dựa trên kiến trúc và lõi ARM. ST Microelectronics là một trong số đó, và nó hiện là nhà sản xuất duy nhất bán danh mục đầy đủ các bộ vi xử lý dựa trên Cortex-M. ARM Holdings không sản xuất hay bán các thiết bị CPU dựa trên thiết kế của riêng mình, mà là cấp phép kiến trúc bộ xử lý cho các bên quan tâm. ARM đưa ra nhiều điều khoản cấp phép, khác nhau về chi phí và sản phẩm phân phối. Khi đề cập đến lõi Cortex-M, người ta cũng thường nói đến Các lõi Sở hữu trí tuệ (IP), nghĩa là bố cục thiết kế chip được coi là trí tuệ tài sản của một bên, cụ thể là ARM Holdings. Nhờ mô hình kinh doanh này và các tính năng thực sự thú vị như khả năng tiêu thụ điện năng thấp, chi phí sản xuất thấp của một số kiến trúc, v.v., ARM là tập lệnh

được sử dụng rộng rãi nhất kiến trúc về mặt số lượng. Các sản phẩm dựa trên ARM đã trở nên cực kỳ phổ biến. Nhiều hơn Tính đến năm 2014, 50 tỷ bộ vi xử lý ARM đã được sản xuất, 10 tỷ trong số đó được sản xuất vào năm 2013. Bộ xử lý dựa trên ARM trang bị cho khoảng 75% thiết bị di động trên thế giới. Rất nhiều chủ đạo và các CPU 64-bit và đa lõi phổ biến, được sử dụng trong các thiết bị đã trở thành biểu tượng trong lĩnh vực điện tử ngành (ví dụ: iPhone của Apple), dựa trên kiến trúc ARM (ARMv8-A).

Là một loại tiêu chuẩn phổ biến, có rất nhiều trình biên dịch và công cụ, cũng như Hệ điều hành (Linux là Hệ điều hành được sử dụng nhiều nhất trên bộ xử lý Cortex-A) hỗ trợ những kiến trúc này, cung cấp cho các nhà phát triển nhiều cơ hội để xây dựng ứng dụng của họ.

1.2.1.1 Bộ xử lý dựa trên Cortex và Cortex-M

ARM Cortex là một tập hợp rộng rãi các kiến trúc và lõi 32/64-bit thực sự phổ biến trong thế giới nhúng. Vi điều khiển Cortex được chia thành ba phân họ chính: • Cortex-A, viết tắt của Application, là một loạt các bộ xử lý cung cấp một loạt các giải pháp cho các thiết bị thực hiện các tác vụ tính toán phức tạp, chẳng hạn như lưu trữ một Hệ điều hành phong phú Nền tảng hệ thống (OS) (Linux và Android phái sinh của nó là những nền tảng phổ biến nhất), và hỗ trợ nhiều ứng dụng phần mềm. Các lõi Cortex-A trang bị cho các bộ vi xử lý được tìm thấy trong hầu hết các của các thiết bị di động, như điện thoại và máy tính bảng. Trong phân khúc thị trường này, chúng ta có thể tìm thấy một số silicon các nhà sản xuất khác nhau, từ những người bán các bộ phận trong danh mục (TI hoặc Freescale) đến những người sản xuất bộ xử lý cho những người được cấp phép khác. Trong số các lõi phổ biến nhất trong phân khúc này, chúng tôi có thể tìm thấy bộ vi xử lý 32-bit Cortex-A7 và Cortex-A9, cũng như hiệu suất cực cao mới nhất 64-bit Cortex-A53 và Cortex-A57 lõi. • Cortex-M, viết tắt của eMbedded, là một loạt các thiết bị có thể mở rộng, tương thích, tiết kiệm năng lượng và bộ vi xử lý dễ sử dụng được thiết kế cho thị trường nhúng chi phí thấp. Họ Cortex-M được tối ưu hóa cho các MCU nhạy cảm về chi phí và điện năng phù hợp với các ứng dụng như Internet of Mọi thứ, kết nối, điều khiển động cơ, đo sáng thông minh, thiết bị giao diện người, ô tô và hệ thống điều khiển công nghiệp, thiết bị gia dụng trong nước, sản phẩm tiêu dùng và y tế dụng cụ. Trong phân khúc thị trường này, chúng ta có thể tìm thấy nhiều nhà sản xuất silicon sản xuất Bộ vi xử lý Cortex-M: ST Microelectronics là một trong số đó. • Cortex-R, viết tắt của Real-Time, là một loạt bộ vi xử lý cung cấp hiệu suất cao các giải pháp tính toán cho các hệ thống nhúng có độ tin cậy, tính sẵn sàng cao, khả năng chịu lỗi, khả năng bảo trì và phản ứng thời gian thực xác định là điều cần thiết. Bộ xử lý dòng Cortex-R cung cấp quá trình xử lý nhanh chóng và xác định và hiệu suất cao, đồng thời đáp ứng các thách thức các ràng buộc thời gian thực. Họ kết hợp các tính năng này trong một hiệu suất, sức mạnh và khu vực được tối ưu hóa gói, làm cho chúng trở thành sự lựa

chọn đáng tin cậy trong các hệ thống đáng tin cậy đòi hỏi khả năng chịu lỗi. Các phần tiếp theo sẽ giới thiệu các tính năng chính của bộ xử lý Cortex-M, đặc biệt là từ quan điểm của nhà phát triển nhúng.

Thanh ghi trong lõi ARM

Giống như tất cả các kiến trúc RISC, bộ xử lý Cortex-M là máy tải / lưu trữ, thực hiện các hoạt động chỉ trên thanh ghi CPU ngoại trừ cho hai loại lệnh: tải và lưu trữ, được sử dụng để chuyển dữ liệu giữa các thanh ghi CPU và các vị trí bộ nhớ.

Hình 2 cho thấy các thanh ghi trong lõi Cortex-M. Một số trong số chúng chỉ có sẵn trong họ hiệu suất cao như M3, M4 và M7. R0-R12 là các thanh ghi mục đích chung và có thể được sử dụng dưới dạng toán hạng cho các lệnh ARM. Tuy nhiên, một số thanh ghi có mục đích chung có thể được sử dụng bởi trình biên dịch như các thanh ghi với các chức năng đặc biệt. R13 là thanh ghi con trỏ ngăn xếp (SP), cũng được cho là được ngân hàng. Điều này có nghĩa là nội dung thanh ghi thay đổi theo chế độ CPU hiện tại (đặc quyền hoặc không đặc quyền). Chức năng này thường được sử dụng bởi Hệ điều hành thời gian thực (RTOS) để thực hiện chuyển đổi ngữ cảnh.

Ví dụ: hãy xem xét đoạn mã C sau bằng cách sử dụng các biến cục bộ "a", "b", "c":

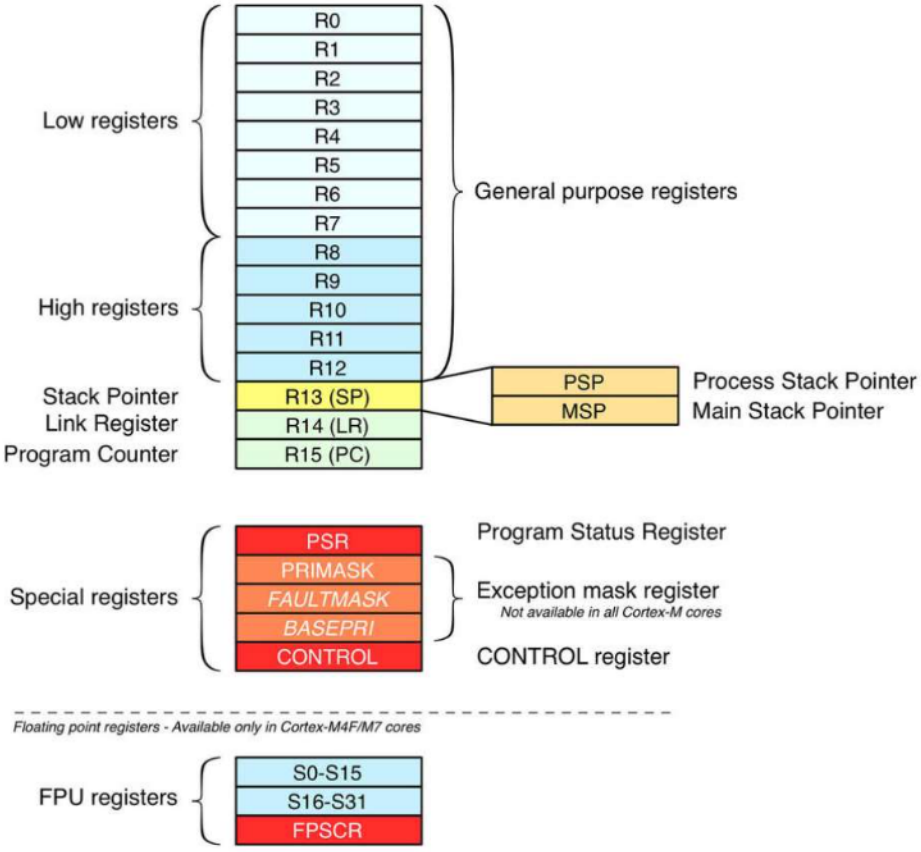
```
...
        uint8_t a,b,c;

        a = 3;
        b = 2;
        c = a * b;
...
```

Trình biên dịch sẽ tạo mã lệnh assembly cho ARM như sau:

```
1  movs     r3, #3       ;move "3" in register r3
2  strb     r3, [r7, #7] ;store the content of r3 in "a"
3  movs     r3, #2       ;move "2" in register r3
4  strb     r3, [r7, #6] ;store the content of r3 in "b"
5  ldrb     r2, [r7, #7] ;load the content of "a" in r2
6  ldrb     r3, [r7, #6] ;load the content of "b" in r3
7  smulbb   r3, r2, r3   ;multiply "a" with "b" and store result in r3
8  strb     r3, [r7, #5] ;store the result in "c"
```

Như chúng ta có thể thấy, tất cả các hoạt động luôn liên quan đến một thanh ghi. Hướng dẫn ở dòng 1-2 chuyển số 3 vào thanh ghi r3 và sau đó lưu trữ nội dung của nó (tức là số 3) bên trong vị trí bộ nhớ được cung cấp bởi thanh ghi r7 cộng với phần bù của 7 vị trí bộ nhớ - đó là nơi lưu trữ một biến. Điều tương tự cũng xảy ra đối với biến b ở dòng 3-4. Sau đó dòng 5-7 tải nội dung của biến a và b và thực hiện phép nhân. Cuối cùng, dòng 8 lưu kết quả vào vị trí bộ nhớ của biến c.



Hình 1.2. Các thanh ghi trong lõi ARM Cortex-M

Tập lệnh Thumb-2

Về mặt lịch sử, các bộ xử lý ARM cung cấp một bộ hướng dẫn 32-bit. Điều này không chỉ cho phép tạo ra một tập hợp phong phú mà còn đảm bảo hiệu suất tốt nhất trong quá trình thực hiện các hướng dẫn liên quan đến các phép toán số học và chuyển bộ nhớ giữa các thanh ghi lõi và SRAM. Tuy nhiên, một tập lệnh 32-bit phải trả giá về dung lượng bộ nhớ của phần firmware. Điều này có nghĩa là một chương trình được viết bằng Kiến trúc tập lệnh (ISA) 32-bit yêu cầu số lượng byte được lưu trữ trong flash cao hơn, ảnh hưởng đến mức tiêu thụ điện năng và chi phí tổng thể của MCU (tấm silicon đắt tiền và các nhà sản xuất liên tục thu nhỏ kích thước chip để giảm giá thành của chúng).

Để giải quyết các vấn đề như vậy, ARM đã giới thiệu tập lệnh Thumb 16-bit, là một tập con của tập lệnh 32-bit thông dụng nhất. Mỗi lệnh dài 16 bit và tự động "được dịch" sang lệnh ARM 32-bit tương ứng có tác dụng tương tự đối với bộ xử lý. Điều này có nghĩa là tập lệnh Thumb 16 bit được mở rộng một cách minh bạch (từ nhà phát triển theo quan điểm) đến các hướng dẫn ARM 32-bit đầy đủ trong thời gian thực mà không làm giảm hiệu suất. Mã Thumb thường là 65% kích thước của mã ARM và cung cấp 160% hiệu suất của mã sau này khi chạy từ hệ thống bộ nhớ 16-bit; tuy nhiên, trong Thumb, các mã lệnh 16-bit có ít chức năng hơn. Ví dụ, chỉ các nhánh mới có thể có điều

kiện và nhiều mã opcodes bị hạn chế chỉ truy cập một nửa của tất cả các thanh ghi mục đích chung của CPU.

Sau đó, ARM giới thiệu tập lệnh Thumb-2, là sự kết hợp của tập lệnh 16 và 32-bit trong một trạng thái hoạt động. Thumb-2 là một tập lệnh có độ dài thay đổi và cung cấp nhiều lệnh hơn so với Thumb, đạt được mật độ mã tương tự.

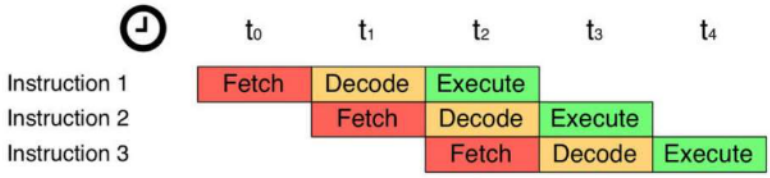
Cortex-M3 / 4/7 được thiết kế để hỗ trợ tập hợp lệnh Thumb và Thumb-2 đầy đủ, và một số chúng hỗ trợ các tập lệnh khác dành riêng cho các hoạt động dấu phẩy động (Cortex-M4 / 7) và hoạt động đơn thao tác đa dữ liệu (SIMD – single instruction multiple data) (còn được gọi là lệnh NEON).

Cơ chế xử lý lệnh Pipeline

Bất cứ khi nào chúng ta nói về việc thực thi các lệnh, chúng ta đang đưa ra một loạt các giả định không hề tầm thường. Trước khi một lệnh được thực thi, bộ xử lý trung tâm (CPU) phải tìm nạp nó từ bộ nhớ và giải mã nó. Quy trình này sử dụng một số chu kỳ CPU, tùy thuộc vào bộ nhớ và kiến trúc lõi CPU, được thêm vào chi phí lệnh thực tế (nghĩa là số chu kỳ cần thiết để thực hiện lệnh).

Các CPU hiện đại giới thiệu một cách để song song hóa các hoạt động này để tăng khả năng xử lý đồng thời các lệnh (số lượng lệnh có thể được thực hiện trong một đơn vị thời gian). Cơ bản chu kỳ hướng dẫn được chia thành một loạt các bước, như thể các hướng dẫn đi dọc theo một đường ống. Thay vì xử lý từng lệnh một cách tuần tự (mỗi lần một lệnh, hoàn thành một lệnh trước bắt đầu với bước tiếp theo), mỗi hướng dẫn được chia thành một chuỗi các giai đoạn để các bước khác nhau có thể được thực hiện song song.

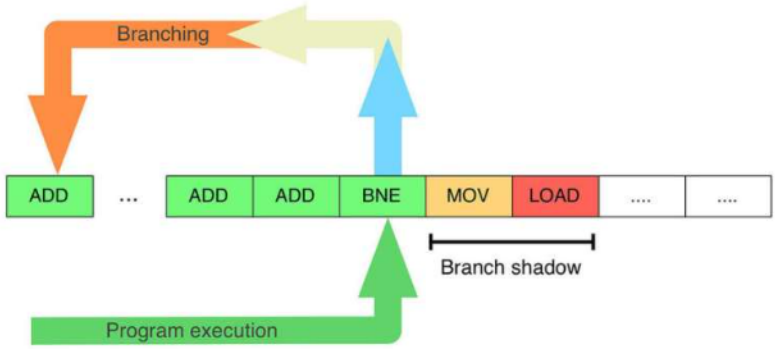
Tất cả các bộ vi điều khiển dựa trên Cortex-M đều giới thiệu một dạng pipelining. Phổ biến nhất là Đường ống 3 giai đoạn, như trong Hình 8. Đường ống 3 giai đoạn được hỗ trợ bởi Cortex-M0 / 3/4. Cortex-M0 + lõi, dành riêng cho MCU công suất thấp, cung cấp một đường ống 2 giai đoạn (mặc dù pipelining giúp giảm chi phí thời gian liên quan đến chu kỳ tìm nạp / giải mã / thực thi của lệnh, nó giới thiệu chi phí năng lượng phải được giảm thiểu trong các ứng dụng công suất thấp). Các lõi Cortex-M7 cung cấp một Đường ống 6 tầng.



Hình 1.6. Cơ chế xử lý lệnh Pipeline

Khi xử lý pipeline, rẽ nhánh là một vấn đề cần được giải quyết. Thực hiện chương trình là tất cả về việc đi các con đường khác nhau; điều này đạt được thông qua sự phân nhánh. Không may, sự phân nhánh gây ra sự vô hiệu của các luồng pipeline, như thể

hiện trong Hình 1.7. Hai lệnh cuối cùng đã được tải vào đường ống nhưng chúng bị loại bỏ do đường dẫn nhánh tùy chọn được sử dụng.



Hình 1.7. Ảnh hưởng của lệnh rẽ nhánh với hoạt động của Pipeline

Ngắt và Xử lý ngoại lệ (Interrupt và Exception)

Ngắt và quản lý ngoại lệ là một trong những tính năng mạnh mẽ nhất của nền tảng Cortex-M bộ vi xử lý. Ngắt và ngoại lệ là các sự kiện không đồng bộ làm thay đổi dòng chương trình. Khi một ngoại lệ hoặc một ngắt xảy ra, CPU tạm ngừng thực hiện tác vụ hiện tại, lưu ngữ cảnh của nó (nghĩa là con trỏ ngăn xếp của nó) và bắt đầu thực hiện một quy trình được thiết kế để xử lý việc ngắt biến cố. Quy trình này được gọi là Trình xử lý ngoại lệ trong trường hợp ngoại lệ và Quy trình dịch vụ ngắt (ISR) trong trường hợp ngắt. Sau khi ngoại lệ hoặc ngắt đã được xử lý, CPU sẽ tiếp tục luồng thực thi trước đó và tác vụ trước đó có thể tiếp tục thực thi[11]. Trong kiến trúc ARM, ngắt là một loại ngoại lệ. Ngắt thường được tạo ra từ thiết bị ngoại vi trên chip (ví dụ: bộ đếm thời gian) hoặc đầu vào bên ngoài (ví dụ: công tắc xúc giác được kết nối với GPIO) và trong một số trường hợp chúng có thể được kích hoạt bởi phần mềm. Thay vào đó, các ngoại lệ liên quan đến việc thực thi phần mềm, và bản thân CPU có thể là một nguồn ngoại lệ. Đây có thể là các sự kiện lỗi chẳng hạn như một nỗ lực truy cập vị trí bộ nhớ không hợp lệ hoặc các sự kiện do Hệ điều hành tạo, nếu có. Mỗi ngoại lệ (và do đó ngắt) có một số nhận dạng duy nhất nó. Bảng 1 cho thấy các ngoại lệ được xác định trước chung cho tất cả các lõi Cortex-M, cộng với một số lượng thay đổi do người dùng xác định những cái liên quan đến quản lý ngắt. Con số này phản ánh vị trí của trình xử lý ngoại lệ quy trình bên trong bảng vectơ, nơi lưu trữ địa chỉ thực của quy trình. Ví dụ, vị trí 15 chứa địa chỉ bộ nhớ của vùng mã chứa trình xử lý ngoại lệ cho SysTick ngắt, được tạo ra khi bộ đếm thời gian SysTick về không.

Bảng 1. Các ngoại lệ của Cortex-M