GIF-3002 Présentation du STM32F407, Logiciel

Ce document présente le jeu d'instruction Thumbs-2 et certains points de l'architecture ARM-Cortex M4. Le but visé est de donner les informations de base afin de réaliser un programme avec le microcontrôleur du cours. Le document présente aussi plusieurs notes concernant l'ensemble des microprocesseurs et microcontrôleurs.

1 Introduction

Les programmes embarqués sont habituellement écrits en C ou en C++ et le compilateur gère la plupart des informations spécifiques à l'architecture du microprocesseur. Dans beaucoup de cas, des routines peuvent être transférées d'un microprocesseur à l'autre, sans impliquer une connaissance intime des instructions supportées par le microprocesseur ou sans connaître le nombre de registres disponibles.

> *La programmation en assembleur de logiciel embarqué est très rare de nos jours. Écrire du code en assembleur est complexe, laborieux et le code n'est pas portable ! D'un autre côté, de très bons compilateurs sont habituellement disponibles sur le marché... La programmation en assembleur devient nécessaire uniquement dans certains cas particuliers: il faut que le code soit très très rapide ou qu'il respecte des temps très précis, il faut que le code gère très étroitement le matériel du système (même dans ce cas, la programmation est souvent en C…), il faut que le code soit très très compact ou il faut déverminer un bogue très nébuleux!*

Cependant, connaître l'architecture des microprocesseurs sur lesquels on travaille est nécessaire dans beaucoup de circonstances, Par exemple, cela est requis si on optimise des parties de programme en assembleur, dévermine du code optimisé, effectue des opérations bas-niveau avec la mémoire, conçoit des systèmes multiprocesseurs, etc. Connaître l'architecture des microprocesseurs sur lesquels on travaille permet aussi de mieux programmer en C/C++ et de mieux exploiter les ressources du microprocesseur.

Pour ces raisons, les sections suivantes décrivent l'architecture du STM32F4, la structure de sa mémoire et sa gestion des interruptions. Cela donne également un exemple d'architecture de microprocesseur qui permettra au lecteur d'extrapoler pour d'autres architectures.

2 Architecture du microprocesseur

Cette section décrit l'architecture ARM-Cortex M4 et donne quelques informations sur les instructions Thumbs-2.

Les instructions présentées seront les instructions requises pour faire un programme élémentaire en assembleur. En effet, quel que soit le langage de programmation employé, il faut savoir : comment transférer des données (modes d'adressage), quelles opérations mathématiques sont possibles, comment faire des boucles et des énoncés conditionnels et comment faire des appels de fonction… Les instructions présentées décriront ces aspects pour l'assembleur Thumbs-2.

2.1 Aspects généraux

Le cœur du STM32F407, un cœur ARM-Cortex M4, est un cœur 32-bits supportant 16 registres. Il a une architecture Harvard (un bus pour les données et un bus pour les instructions) et un modèle Load-Store pour accéder aux données de la mémoire. Le ARM-Cortex M4 supporte le jeu d'instruction Thumbs-2 et les instructions peuvent être exécutées sur des variables 8bits (octet), 16bits (demi-mot) ou 32bits (mot). Le STM32F407 opère en deux modes : le mode Thread et le mode Handler. Enfin, le cœur contient des interfaces pour observer et contrôler l'exécution d'instructions (pour déverminer le code!!!)(JTAG, SWD et Trace MacroCell).

En plus de ces caractéristiques, le cœur inclut quelques périphériques et interfaces spécifiques : un contrôleur d'interruption (NVIC = Nested Vectored Interrupt Controller), un bloc de contrôle du cœur (SCB =System Control Block), un timer (le systick timer), une unité de protection de la mémoire (MPU = Memory Protection Unit) et une unité de traitement des fractions (FPU = floating point unit).

Microprocesseur 32-bits

Le ARM-Cortex M4 est un microprocesseur 32-bits : ses registres ont 32 bits, il peut adresser 32-bits de mémoire (2^32 = 4Go), ses bus sont des bus ayant 32 bits de données, ses instructions ont 16 ou 32 bits… bref le STM32F407 est un microprocesseur 32 bits.

Registres 32-bits

Le ARM-Cortex M4 a 13 registres tout usage (R0 à R12), une banque de deux registres pour la pile (SP_process et SP_Main --- R13), un registre de lien (R14) et un registre Program Counter (R15). En plus de ces registres, le ARM-Cortex M4 a une banque de registres pour l'état du programme.

> *La pile est une structure de donnée, à l'intérieur de la mémoire de données. Son rôle principal, dans les systèmes embarqués, est de sauvegarder temporairement des informations telles que l'adresse de retour des fonctions, après un appel de fonction.*
>
> *Lorsqu'un registre ou une mémoire est "banked", il y a plusieurs copies matérielles de ce registre ou de cette mémoire. Lors de l'exécution d'un programme, une seule copie est utilisée et les autres copies sont masquées, invisibles. Habituellement, un*

registre de configuration permet de choisir la copie active parmi les copies de la banque. Les banques de registre ou de mémoire sont très utiles lors de changements de contexte : elles permettent de changer les variables utilisées avec très peu d'opérations.

Le registre de lien sert lors d'appel de fonction. Lorsqu'une fonction est appelée, le registre de lien contiendra l'adresse de retour de la fonction.

Le Program Counter indique l'adresse de la prochaine instruction à exécuter.

Les registres d'états indiquent principalement la valeur des drapeaux de l'ALU, c'est-à-dire qu'ils donnent des informations sur la dernière opération mathématique effectuée. Ils contiennent aussi des informations sur le mode d'opération et les interruptions.

Architecture Harvard

Le ARM-Cortex M4 a un bus système pour les données et pour les périphériques. Ce bus est connecté à une matrice d'interconnexion qui le relie à la mémoire SRAM et aux périphériques. Il a aussi deux bus pour les instructions (ICode et DCode) qui permettent de charger (fetch) les instructions (ICode) ou de lire/écrire la mémoire FLASH (DCode).

Dans tous les systèmes microprocesseurs, la mémoire d'instruction (ROM/FLASH/EEPROM/…) contient aussi des données : toutes les constantes ou tables de constantes, des paramètres d'opérations gérés par l'application, et, potentiellement, d'autres informations propres au système microprocesseur (la version du produit par exemple!!!). Pour lire et exécuter une instruction, il est donc possible de lire la mémoire d'instruction deux fois…

Architecture Load-Store

Le ARM-Cortex M4 a une architecture Load/Store: seules les instructions de LOAD et de STORE permettent d'accéder aux données ou aux périphériques. Les opérations mathématiques s'effectuent toutes sur des registres et ne peuvent avoir une adresse de mémoire comme paramètre.

Presque tous les microprocesseurs modernes ont des architectures Load/Store, par opposition à certaines architectures Registre/Memoire qui comprenant des instructions plus complexes faisant à la fois des accès mémoire et des opérations mathématiques. D'un point de vue architecture des microprocesseurs, les architectures Load/Store sont beaucoup plus simples et cette simplicité devient nécessaire lorsque l'architeture du microprocesseur devient plus élaborée. Par exemple, un pipeline dans une architecture load/store est beaucoup plus facile à gérer que dans une architecture Registre/Memoire.

Instructions Thumbs-2
Le ARM-Cortex M4 supporte le jeu d'instruction Thumbs-2. Ce jeu d'instruction est un hybride entre une architecture RISC et CISC : il comprend des instructions sur 16 bits et des instructions sur 32-bits.

> *RISC signifie Reduced Instruction Set Computer. Dans une architecture RISC, toutes les instructions ont la même longueur. Elles sont simples, faciles à gérer et elles s'exécutent rapidement. CISC signifie Complex Instruction Set Computer. Daus un architecture CISC, les instructions ont des longueurs variables. Elles peuvent être complexes et plus dures à gérer. En contrepartie, un programme CISC est plus dense qu'un programme RISC.*
>
> *Les microprocesseurs ARM supportent habituellement deux jeux d'instructions: le ARM (instructions RISC, 32bits) et le Thumbs (instructions RISC, 16bits). Le microprocesseur inter opère entre les deux jeux d'instructions de façon fluide.*

Format des données
Bien que le microprocesseur soit 32-bits (un mot est sur 32 bits), il peut manipuler des variables plus petites afin d'économiser de l'espace mémoire.

Modes d'opération
Les modes d'opérations du ARM Cortex sont des modes pour supporter un système d'exploitation. Le mode Thread est le mode par défaut, qui s'exécute au démarrage du microprocesseur. Le mode Handler est le mode d'opération qui s'exécute lorsqu'une interruption se produit.

JTAG, SWD et Trace
Matériel permettant de suspendre l'exécution d'instructions, d'exécuter des instructions pas-à-pas, lire les registres du microprocesseur, accéder aux bus des mémoires et plus encore. Permet le debug du code exécuter par le cœur.

Nested Vectored Interrupt Controller
Le NVIC est une interface entre les nombreux signaux d'interruption des périphériques et les moins nombreux signaux d'interruption du cœur. Il sert à gérer les séquences d'interruptions, la priorité de chaque interruption, les vecteurs d'interruptions, et plus…

System Control Block
Le bloc de contrôle du système contient plusieurs registres qui permettent au programmeur de contrôler l'opération du processeur : mode d'opération (économie de puissance), options de démarrage, alignement des mots dans la mémoire, gestion des exceptions, et plus…

Systick Timer
Minuterie 24 bits très simple servant souvent à générer les interruptions du système d'exploitation.

Memory Protection Unit
Circuit matériel optionnel validant les accès à la mémoire lorsqu'ils sont fait par les programmes usager (en dehors des interruptions).

Floating Point Unit
Matériel pemettant de faire des additions, des soustractions, des multiplications, des divisions et différentes autres opérations arithmétiques impliquant des fractions.

2.2 Registres

Le Cortex M4 a les registres illustrés ci-dessous[1] :

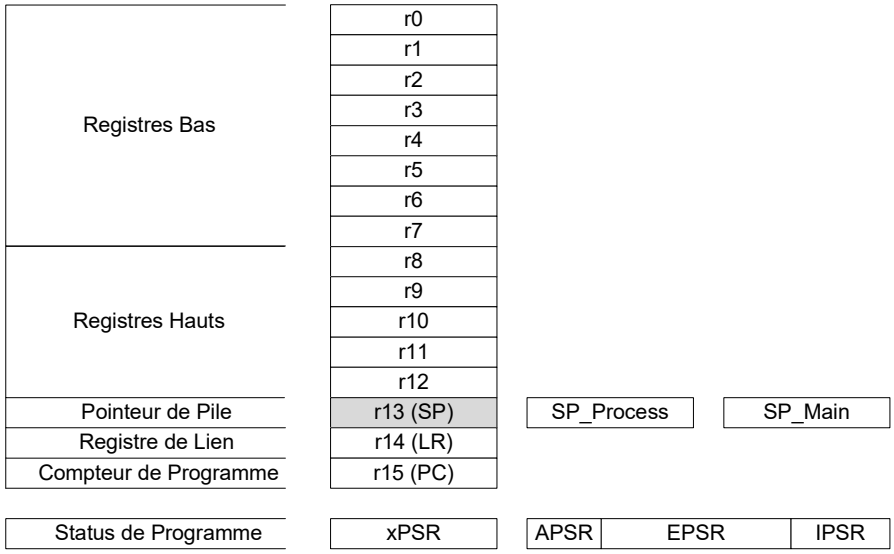

Figure 1 - Registres du ARM-Cortex M4

Les Registres Bas (r0 à r7) sont des registres tout usage disponibles pour toutes les instructions, quelles soient encodées sur 16 bits ou sur 32 bits.
Les registres hauts sont aussi des registres tout usage disponibles uniquement pour les instructions 32 bits.

> *Il faut 3 bits pour désigner un registre parmi les registres bas. Il faut 4 bits pour désigner un registre parmi les 16 registres de r0 à r15!*

Le pointeur de pile indique l'adresse de la mémoire étant le haut de la pile. Il y a deux registres de pile : SP_Process et SP_Main. Le microprocesseur choisit automatiquement un de ces registres de la banque de registre en fonction du mode d'opération et de registres de configuration : lors d'une interruption, c'est-à-dire en mode "handler" ou du système d'exploitation, SP_Main est toujours utilisé. En dehors des interruptions,

[1] Voir la section *2.3 Registers* du document *CortexM3_TRM_r2p0.pdf* pour plus de détails; Ce document est disponible sur le CD-ROM accompagnant le kit d'évaluation du LM3S9B92

SP_Main est aussi utilisé par défaut, à moins qu'un éventuel système d'exploitation ait changé la configuration du microprocesseur pour que SP_Process soit utilisé.

Pour changer de contexte, c'est-à-dire passer d'un programme usager à un autre programme usager, ou pour passer d'un programme usager à une routine du système d'exploitation, il est nécessaire d'utiliser une pile différente par contexte. Avoir une banque de pointeurs de pile facilite l'implémentation d'un système d'exploitation.

Le registre de lien est automatiquement mis à jour lors d'appel de fonction : il contient l'adresse de retour de la fonction (l'adresse de l'instruction à être exécutée lorsque l'exécution de la fonction sera terminée).

Le registre PC est le compteur de programme. Ce registre indique l'adresse des instructions à être exécutées. Changer r15 équivaut à faire un saut à travers le code du programme.

En raison du pipeline d'instruction du ARM Cortex M4, PC désigne plusieurs instructions.

Enfin, le registre xPSR est un registre indiquant l'état de l'exécution d'un programme (Program Status Register). Il peut indiquer la valeur des drapeaux de l'ALU (APSR), l'état de l'exécution (EPSR) ou le numéro de l'interruption en cours (IPSR).

Le registre xPSR ne fait pas partie des 16 registres pouvant être adressé par les instructions arithmétiques du microprocesseur. Il faut utiliser des instructions spéciales pour lire ou écrire ce registre.

APSR contient 5 drapeaux : Overflow, Carry, Negative, Zero, Sticky Saturation. Overflow indique que le résultat déborde des valeurs premises. Carry indique un emprunt ou une retenue lors d'une addition, Negative indique si le résultat est négatif, Zero indique si le résultat est zéro et Sticky saturation indique le résultat a saturé (certaines instructions demandent de ne pas faire de roll-over quand les minimums et maximums sont atteints) ou lorsque le résultat de multiplication déborde.

EPSR indique si le microprocesseur exécute présentement une instruction If-Then ou une instruction Load/Store Multiple Register. Cette connaissance est nécessaire parce que ces instructions sont traitées spécialement dans la pipeline d'instruction du microprocesseur lorsque survient une interruption.

IPSR indique le numéro du l'interruption en cours.

2.3 Mémoire et modes d'adressage

2.3.1 Memory Map

Le cœur ARM Cortex M4 supporte des adresses sur 32 bits. Les 4Go adresses possibles sont réparties entre la mémoire non-volatile (FLASH), la mémoire volatile interne et

externe (RAM), les périphériques, les registres propres au microprocesseur lui-même, des adresses réservées au périphériques du vendeur et plus. Par défaut, un mot de mémoire est sur 8 bits.

Le cœur ARM Cortex M4 est un cœur Memory Mapped Input Output : les adresses des périphériques sont dans le même espace d'adresses que la mémoire. Cela simplifie énormément l'accès à toutes les composantes du système et facilite l'encodage des instructions.

Ainsi un manufacturier voulant intégrer un cœur ARM Cortex M4 dans un microcontrôleur qui lui est propre pourra également mettre jusqu'à 512MB de FLASH, 512MB de mémoire RAM interne, 1GB de mémoire RAM externe, plusieurs périphériques avec des adresses pour leurs registres et plus encore…

La section *2.2* sur le *Memory Map* du document *DM00046982.pdf* présente le Memory Map du ARM Cortex M4. Vous retrouverez plus de détails dans cette section.

La carte mémoire du STM32F407 ci-dessous (voir la section 3) illustre l'intégration d'un cœur ARM Cortex M4 à l'intérieur d'un microcontrôleur.

2.3.2 Modes d'adressage

2.3.2.1 Mémoire/Registre

Tous les accès à la mémoire se font via des instructions LOAD ou STORE.

Les instructions LOAD et STORE ont, pour paramètres, un numéro de registre qui sera la destination (LOAD) ou la source (STORE) et une adresse de mémoire (pouvant être désignée par un registre).

Il existe plusieurs façons de déterminer l'adresse de la mémoire visée par l'instruction. L'adresse peut être déterminée par :

- L'adresse de l'instruction courante + une constante (PC + immediate)
- Le dessus de la pile + une constante (SP + immediate)
- La valeur d'un registre + une constante (indirect indexed)
- La somme de deux registres

Des instructions 32-bits permettent d'utiliser des constantes plus grandes afin de déterminer l'adresse ou, encore, d'incrémenter automatiquement un registre en plus de l'accès à la mémoire.

Comme les instructions sont encodées sur 16 ou 32 bits, il est impossible de faire un load/store à une adresse absolue de la mémoire (adresse sur 32-bits). Il n'y a pas assez de bits dans les instructions pour spécifier 2^32 adresses... Afin d'accéder à une adresse absolue, il faut d'abord mettre la valeur de cette adresse dans un registre.

2.3.2.2 Registre/Registre

Les instructions de type MOV permettent d'affecter la valeur d'un registre à un autre registre. Ces instructions sont simples et elles sont habituellement codées sur 16 bits, mais des MOV plus puissants, permettent de faire des opérations additionnelles sur 32bits.

2.3.2.3 Constante/Registre

Les instructions de type MOV permettent d'affecter une constante à un autre registre.

> *Comme les instructions sont encodées sur 16 ou 32 bits, il est impossible d'affecter de très grandes constantes. Par exemple, il n'y a pas assez de bits dans les instructions faire MOV R0, 0x12345678... Afin de charger de grandes valeurs dans les registres, les constantes sont mises dans la mémoire d'instructions et chargées dans les registres avec un LOAD relatif au PC. Ainsi, a = 0x12; ne sera pas compilé comme a = 0x12345678;*

2.4 Instructions arithmétiques

Toutes les instructions mathématiques se font sur des registres entiers.

Les instructions arithmétiques peuvent avoir trois opérandes : ADD R0, R1, R2. Dans cette exemple, R0 = R1 + R2.

La plupart des instructions mathématiques se font en un cycle d'horloge (habituellement les multiplications et les divisions sont plus longues).

2.5 Instructions de contrôle

Les instructions de contrôle changent la valeur du PC (r15): elles contrôlent la séquence d'exécution des instructions. Entre chaque instruction du microprocesseur, le PC est normalement incrémenté de 2 ou 4 (les instructions sont sur 16 bits ou 32 bits), cependant les instructions de contrôlent interrompent le cours linéaire du programme en changeant PC autrement.

> *Il est possible de lire ou d'écrire le registre PC avec des instructions d'affectation ou des instructions arithmétiques au même titre que pour les autres registres. Cependant, le programmeur ou le compilateur doit tenir compte du pipeline d'instruction du microcontrôleur qui peut changer la valeur du PC pendant l'exécution d'une instruction.*

Il y a quatre catégories d'instructions de contrôle : les sauts directs inconditionnels (**Sauts** --- JMP--- Goto --- Appel de fonction), les sauts directs conditionnels (**Branchement conditionnels**---If), les sauts indirects inconditionnels (**Appel et Retour de fonction**) et les sauts indirects conditionnels.

2.5.1 Sauts (B)

Un saut direct est un saut dont l'adresse est spécifiée dans l'instruction. Ces sauts sont simplement une affectation du registre PC qui tient compte du pipeline d'instruction. Les sauts, avec l'instruction B (Branch) sont relatifs à la valeur actuelle du PC : ils permettent de sauter à une instruction étant à ±16MB par rapport au PC actuel.

2.5.2 Branchements conditionnels (CB)

Les sauts conditionnels servent à implémenter les instructions "if". Le branchement est pris si une condition est rencontrée.

Les conditions possibles sont diverses. On retrouve $R_x == 0$ ($R_x != 0$) ou encore un saut en fonction des drapeaux de l'unité arithmétique et logique (ALU).

La destination du branchement est relative au PC actuel.

2.5.3 Appel et retour de fonctions (BL, BX, BLX)

Pour appeler une fonction, un saut direct est utilisé, avec l'instruction BL (Branch and Link). L'instruction BL met l'adresse de retour dans le registre R14 (Link Register) et effectue un saut à l'adresse de la procédure appelée (relative au PC, spécifié dans l'instruction).

Pour revenir de la fonction, l'instruction BX R14 est utilisée. L'instruction BX (Branch and eXchange) permet de faire un saut à une adresse absolue spécifiée dans un registre. Comme R14 contient l'adresse de retour (à moins d'avoir été modifié!!!), BX R14 est équivalent à l'instruction return.

L'instruction BLX met l'adresse de retour dans le registre de lien (R14) et met le PC égal au registre désigné par l'instruction. Cette instruction permet d'appeler une fonction avec un pointeur de fonction…

2.6 Encodage des instructions

La section 5.2 de DDI0403D_arm_architecture_v7m_reference_manual.pdf présente les details de l'encodage des instructions Thumbs 2. Il n'est pas nécessaire de mémoriser cet encodage dans le cadre du cours, mais comprendre l'encode des instructions, de manière générale, sera présenté dans le cours.

3 Carte de la mémoire

Le STM32F407 a la carte de mémoire suivante (memory map) :

Memory MAP du ARM-Cortex M4	
Adresse	**Contenu**
0xFFFFFFFFF	0.5GB --- Périphérique du cœur
0xE00000000	
0xD00000000	2GB --- Mémoires et périphériques externes
0xC00000000	
0xA00000000	
0x800000000	
0x600000000	
0x400000000	0.5GB --- Périphériques du manufacturier
0x200000000	0.5GB --- Mémoire RAM interne
0x000000000	0.5GB --- Mémoire code

Figure 2 – Carte mémoire générique d'un microcontrôleur avec ARM-Cortex M4

RAM interne	
Adresse	**Contenu**
0x3FFFFFFF	Réservée
0x20020000	
0x2001FFFF	16KB de SRAM avec bit-banding
0x2001C000	
0x2001BFFF	112KB de SRAM avec bit-banding
0x20000000	

Mémoire code	
Adresse	**Contenu**
0x1FFFFFFF	Réservée
0x1FFFC008	
0x1FFFC007	Option Bytes
0x1FFFC000	
0x1FFFBFFF	Réservée
0x1FFF7A10	
0x1FFF7A0F	Mémoire Système et OTP
0x1FFF0000	
0x1FFEFFFF	Réservée
0x10010000	
0x1000FFFF	CCM Data RAM
0x10000000	
0x081FFFFF	Reservée
0x08100000	
0x080FFFFF	FLASH
0x08000000	
0x07FFFFFF	Réservée
0x00100000	
0x000FFFFF	Alias de FLASH, SRAM ou Mémoire Système
0x00000000	

Figure 3 – Carte mémoire du STM32F407

Voici plusieurs observations sur cette carte mémoire :

- Il y a 4Go adresses, soit 2^{32}.
- Il s'agit d'un système MMIO (Memory Map IO) : les périphériques et les mémoires partagent le même espace et ils ont des adresses distinctes.
- Le STM32F407 n'utilise qu'une fraction des adresses que supporte le cœur du microcontrôleur. De larges espaces d'adresses du cœur sont inutilisés.
- Il y a trois plages de SRAM distinctes pour les trois SRAMs du microcontrôleur.

Il y a plusieurs mémoires SRAM dans le système pour plusieurs raisons. D'abord, la mémoire CCM Data RAM (64K) est très utile pour accélérer l'exécution d'instruction : elle permet d'éviter des stalls de pipeline causés par deux accès simultanés à la même mémoire. Ensuite, il y a deux mémoires SRAM pour faciliter les transferts par DMA et éviter l'engorgement de la mémoire, surtout lorsque des périphériques rapide (USB, Ethernet) utilisent le DMA pour mettre des données en mémoire.

- Il est possible de désigner individuellement chaque bit de la mémoire RAM interne à l'aide d'adresses prédéterminées : il s'agit du bit banding. Cela facilite les opérations sur des bits ou sur des variables booléennes.

- Un alias d'adresse signifie que deux adresses (ou plus) désignent le même mot de mémoire. Les adresses les plus basses désignent soit la mémoire SRAM, soit la mémoire FLASH, soit la mémoire système en fonction des broches de boot du microcontrôleur. Il est possible d'accéder à la FLASH ou à la SRAM ou à la mémoire système en utilisant d'autres adresses.

Le décodeur d'adresse du microcontrôleur « décode » l'adresse en fonction de signaux externes au microcontrôleur. Cela permet de décider quelle mémoire contiendra les premières instructions exécutées par le microcontrôleur ainsi que la table des vecteurs d'interruption initiale…

- La mémoire Système et la mémoire OTP sont deux mémoires non-volatiles qui ne peuvent être effacées. La mémoire système contient du code permettant de reprogrammer la FLASH du microcontrôleur (par le UART, le CAN, et plus). La mémoire OTP permet à l'usager d'entreposer des informations permanentes dans le microcontrôleur comme un numéro de série.
- Les bytes d'options (Option Bytes) sont des octets reliés au contrôle de l'alimentation de l'appareil (brown-out, watchdog, modes de sommeil et plus).

4 Interruptions

4.1 Aspects généraux

Le STM32F407 utilise le NVIC d'ARM-Cortex M4 pour gérer les interruptions. Ce Nested Vector Interrupt Controller gère les interruptions (exceptions est employé ici comme un synonyme d'interruptions).

> *Dans la littérature, interruption désigne habituellement un signal provenant d'un périphérique et exception désigne habituellement une faute survenant lors de l'exécution d'une instruction. Cependant, comme une interruption survient de manière exceptionnelle et comme les exceptions interrompent l'exécution en cours, les deux termes sont souvent mélangés.*

4.2 Nested Vectored Interrupt Controller

Lorsqu'une interruption se produit, mais qu'elle ne peut pas être traitée (parce qu'une autre interruption de priorité supérieure est en traitement par exemple), l'interruption est mise de côté et elle est traitée ultérieurement (quand le traitement de l'interruption haute-priorité est terminé par exemple). Le mot anglais pour qualifier une interruption mise de côté est "Nested". La meilleure traduction semble être "imbriquée".

Dans plusieurs microprocesseurs modernes, lorsqu'une interruption doit être traitée immédiatement après une autre interruption, le microprocesseur évite de revenir au programme principal. Il évite de faire certaines étapes préalable au traitement de l'interruption (sauvegarde l'adresse de retour au main par exemple) et il précharge le vecteur d'interruption afin d'accélérer le temps de traitement des interruptions.

Une interruption est dite vectorisée (vectored) quand l'adresse de la routine traitant l'interruption peut être changée : lors de l'interruption le microprocesseur utilise un vecteur (un pointeur de fonction) afin de faire un saut à la séquence d'instructions traitant l'interruption.

4.3 Types et priorités des exceptions

La table d'interruption du STM32F407 sera présentée en cours (voir la table 45 du Reference Manual - DM00031020.pdf*)*. Cette table contient une description des interruptions du STM32F407 ainsi que leur priorité.

> *Une interruption est dite asynchrone lorsqu'elle se produit aléatoirement dans le temps, c'est-à-dire sans aucun lien avec le programme exécuté. Une interruption est dite synchrone lorsqu'elle est liée à la séquence d'instruction exécutée : elle se reproduira si on exécute la séquence d'instruction de nouveau.*
>
> *Une interruption est dite précise si toutes les instructions qui précèdent l'interruption sont terminées et si celles qui suivent n'ont pas encore produit de résultat. Une*

interruption est dite imprécise si le retour de l'interruption peut causer une erreur dans la séquence d'instruction qui aurait été exécutée linéairement sans l'interruption.

4.4 Entrées dans une interruption

Lors d'une entrée dans une interruption, le microprocesseur entreprend les actions suivantes :

1. PUSH des 8 registres suivant sur la pile sélectionnée : xPSR, PC, LR, r12, r3, r2, r1, et r0.
    a. Si les fractions sont mises sur la pile, la taille de la trame empilée lors de l'entrée dans l'interruption augmente.
    b. Si une interruption haute priorité interrompt une interruption basse priorité pendant son entrée et que le PUSH est déjà fait, l'opération n'est pas répétée (voir ci-dessous).
2. Lire la table des vecteurs d'interruptions
    a. La position de la table des vecteurs d'interruption est l'adresse 0 par défaut (au reset), mais elle peut être changée en écrivant un registre du STM32F407.
    b. Le numéro du vecteur d'interruption est donné par "La table d'interruption de la datasheet du STM32F407 » et la position de l'adresse de ISR pour l'interruption dans la table est 4*#vecteur d<interruption
3. Lors de l'interruption de Reset seulement, mettre à jour la valeur initiale de SP
    a. La valeur initiale de SP est emmagasinée à l'index 0 de la table des vecteurs d'interruptions.
4. Mettre à jour le PC avec l'adresse du vecteur d'interruption
5. Décoller le pipeline d'instruction à partir du nouveau PC
6. Mettre à jour le registre de lien (R14, LR) avec l'adresse de retour de l'interruption

4.5 Sortie d'une interruption

Le STM32F407 ne supporte pas d'instruction spécifique pour retourner d'une interruption. En fait, le retour d'une interruption se fait en mettant PC égal à 0xFFFFFFF. Cela peut se faire avec plusieurs instructions : POP/LDM, LDR et BX.

4.6 Latence des interruptions et optimisations

Lorsqu'une interruption se produit, il s'écoule un certain temps avant le traitement de l'interruption : il s'agit de la latence de traitement des interruptions. Dans certains systèmes, le temps de réponse à certains évènements doit être terriblement court et les concepteurs du cœur ARM ont cherché à réduire le temps de latence le plus possible. Ils ont employé les techniques suivantes :

1. Effectuer le plus de tâches possible en parallèle lors de l'entrée dans l'interruption

    a. Par exemple, les valeurs sauvegardées sur la pile peuvent être écrites sur le bus système pendant que la table des vecteurs d'interruptions est lue sur le bus d'instructions.
2. Effectuer automatiquement, par le microprocesseur, quelques opérations omniprésentes lors de l'entrée dans une interruption : sauvegarde sur la pile de certains registres et des drapeaux en plus de l'adresse de retour.
    a. Dans la plupart des systèmes, le programmeur ou le compilateur doivent obligatoirement exécuter quelques instructions au début de chaque interruption. Ce n'est pas nécessaire avec le Cortex M4.
3. Éviter d'effectuer certaines tâches lors de l'entrée dans l'interruption si une autre interruption était en traitement (tail-chaining --- enchaînement de queue ?!?).
    a. Si deux interruptions se suivent queue à queue, il n'est pas nécessaire de sauvegarder deux fois les registres du main sur la pile.
4. Si une interruption haute priorité survient lorsqu'une interruption basse priorité commence à être traitée, mais n'est pas encore exécutée, la séquence d'entrée dans les interruptions n'est pas reprise au complet (Late-arriving --- Entrée tardive).

On retrouve aussi plusieurs techniques reliées aux instructions elles-mêmes:

- *La première chose que fait le microcontrôleur lorsqu'une interruption se produit est de terminer le ou les instructions en cours* : Toutes les instructions qui prennent plusieurs cycles à exécuter (Load multiple ou Store multiple par exemple) ou qui requièrent un traitement particulier lors de l'exécution en pipeline (affectation conditionnelle par exemple) peuvent être « interrompues » par une interruption.
- *Le LOAD MULTIPLE et le STORE MULTIPLE permettent de sauvegarder rapidement les registres sur la pile.* Ainsi, une interruption qui utilise beaucoup de registres pourra sauvegarder l'information de la tâche interrompue avec un minimum d'instructions.

L'excellent site https://community.arm.com/docs/DOC-2607 donne plus de détails sur la latence des interruptions du cortex M4.