Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes Embarqués 1 & 2

p.01 - Interruptions (1re partie)

Classes T-2/I-2 // 2017-2018

Daniel Gachet | HEIA-FR/TIC

p.01 | 21.02.2018

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz

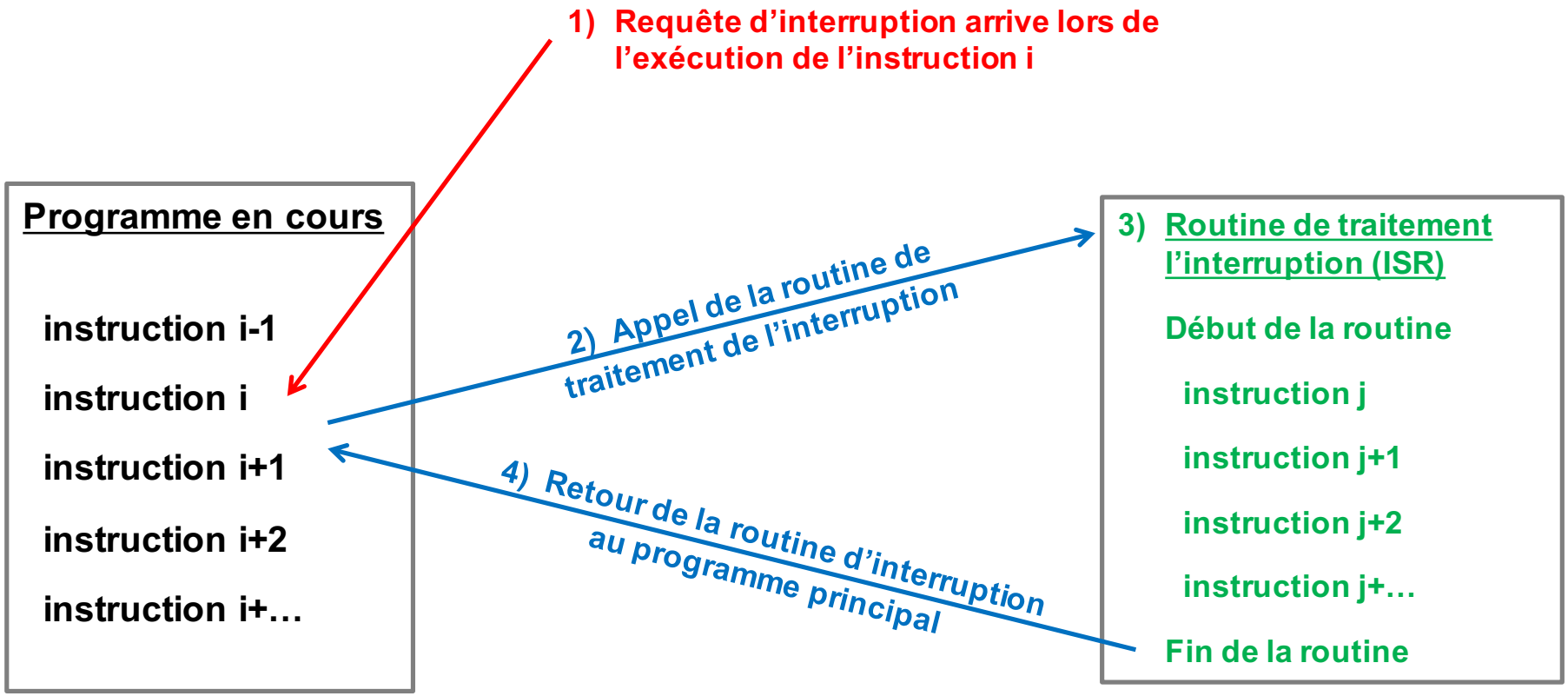
Contenu

- Concept général
- Interruptions
- Séquence d'interruption
- Latence et gigue
- Vecteur d'interruption
- Modes de fonctionnement du processeur
- Entrée et sortie d'une routine d'interruption

Concept général

- Définition (selon le Larousse en ligne http://www.larousse.fr/dictionnaires/francais/interruption)
  Suspension provisoire de l'exécution d'un programme au profit d'un autre.
- Une interruption *(interrupt)* peut être vue comme l'appel d'une fonction déclenché par un signal
- Ce signal est lui-même la conséquence d'un événement, qui peut être interne au programme et résultant de son exécution, ou bien extérieur de ce dernier et indépendant de son exécution

Exemple…


1) Requête d'interruption arrive lors de l'exécution de l'instruction i
Programme en cours
instruction i-1
instruction i
instruction i+1
instruction i+2
instruction i+…
2) Appel de la routine de traitement de l'interruption
3) Routine de traitement l'interruption (ISR)
Début de la routine
instruction j
instruction j+1
instruction j+2
instruction j+…
Fin de la routine
4) Retour de la routine d'interruption au programme principal

Types d'interruptions

- Les interruptions/exceptions peuvent être classées en 2 grandes catégories
  - Les interruptions logicielles *(software interrupts)*
  - Les interruptions matérielles *(hardware interrupts)*

Interruptions logicielles

- Les interruptions logicielles ont trois fonctions principales
  - Permettre à une application logicielle de communiquer avec un système élémentaire d'entrée/sortie (p. ex. BIOS), lequel est contenu dans une mémoire morte d'un ordinateur *(firmware, silicon software)*
  - Permettre à une application logicielle de communiquer avec un système opératif évolué (Linux, Windows,...)
  - Permettre à des outils de développement de poser des points d'arrêt facilitant le débogage de l'application *(software breakpoints)*
- L'instructions des processeurs ARM
  - SVC #n_24 (appel système - syscall)
  - BKPT #n_16 (débogage)

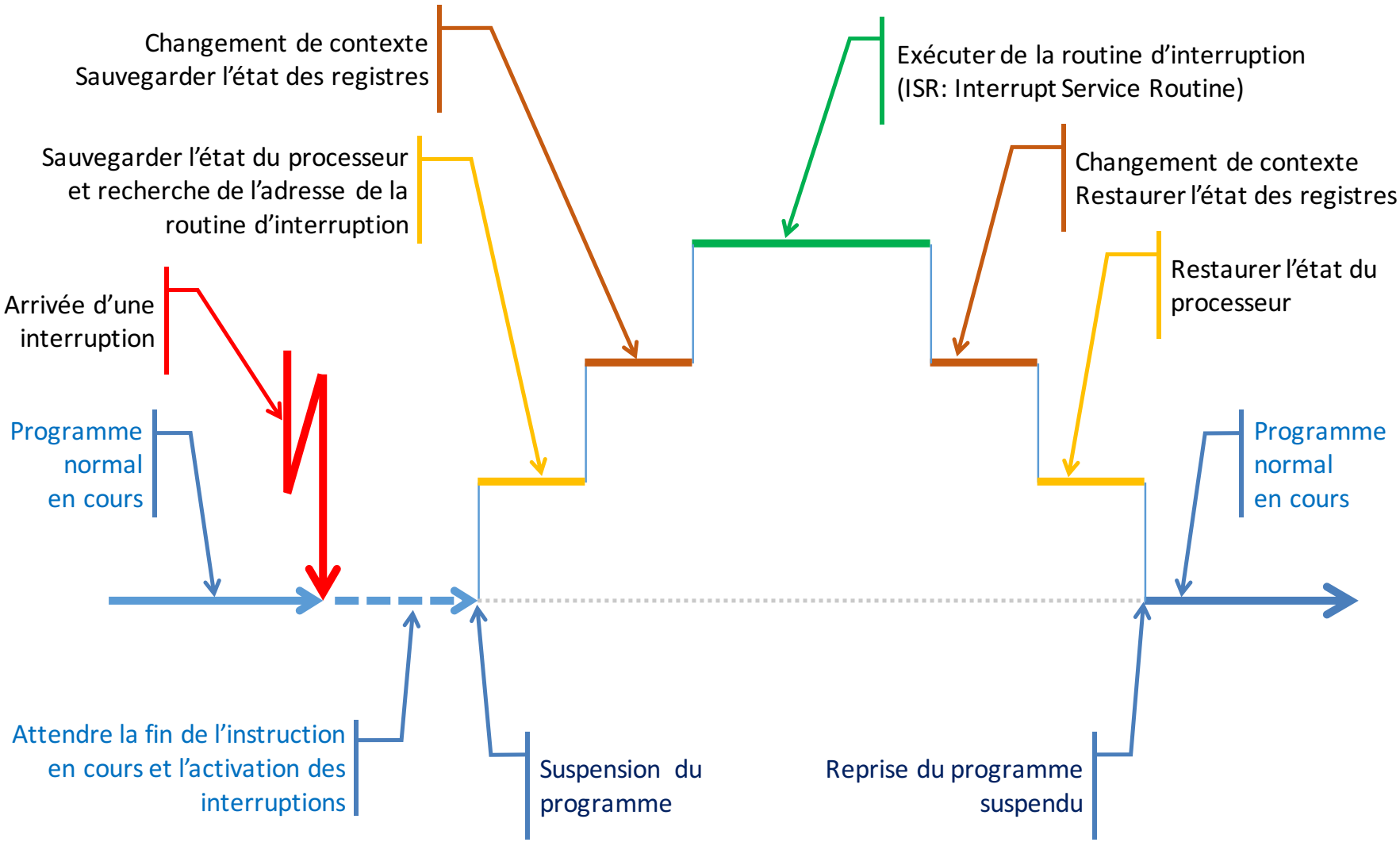
Interruptions matérielles

- Fonction principale
  - Dans le cadre d'échange d'information avec des périphériques ayant un comportement non prévisible/aléatoire (p. ex. cartes réseau, ports séries,...) ou d'urgence (p. ex. signaux d'alarmes), il est souhaitable que les périphériques eux-mêmes puissent prendre l'initiative de l'échange, en forçant le microprocesseur d'interrompre immédiatement l'exécution du programme en cours pour traiter leurs requêtes.
- Fonction auxiliaire
  - Lors de développement d'applications logicielles, il est parfois utile de pouvoir stopper l'exécution du programme en surveillant le bus de données ou d'adresses *(hardware breakpoints, watchpoints)*

Exceptions

- Défaillances et dysfonctionnements
  - Lors de l'exécution d'un programme des erreurs peuvent survenir et perturber son bon déroulement. Les µP implémentent tout un arsenal de mécanismes de reconnaissance d'exceptions et fournissent au logicielle une série de vecteurs spécifiques permettant leur traitement.
- Exceptions logicielles
  - Instruction illégales : instruction pas supportée/implémentée
  - Violation de privilèges : accès à des ressources non autorisées (instructions, zones mémoires,...)
- Exceptions matérielles
  - Reset
  - Erreur sur le bus de données *(bus error)*, p. ex. périphériques ne répondant pas aux cycles d'accès dans les temps spécifiés
  - Erreur sur le bus d'adresses *(address error)* p. ex. accès mémoires mal-alignés *(miss aligned)*

Séquence d'interruption

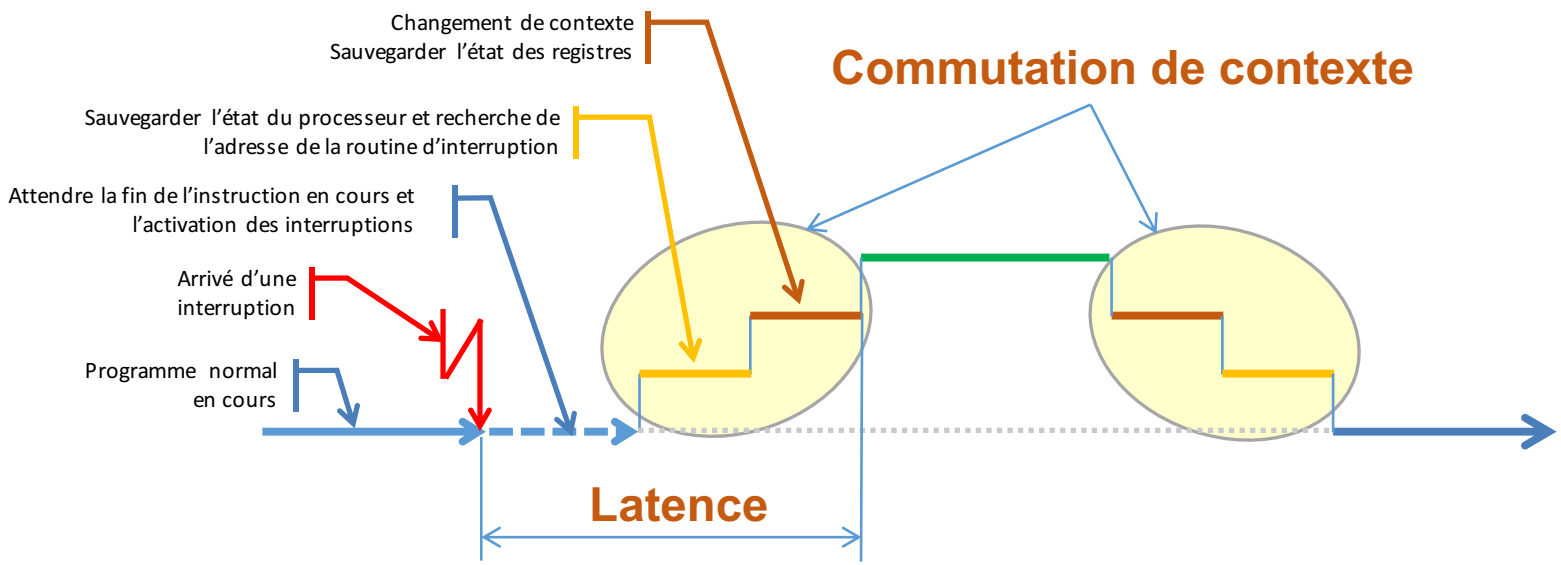


Séquence d'interruption (II)

- Le µP peut autoriser ou bloquer les interruptions par l'intermédiaire de fanions dans son registre d'état *(Program Status Register)*
- Si les interruptions sont autorisées et qu'une d'elles sont levées, les opérations suivantes sont exécutées
  - le µP suspend le programme en cours, sauve l'état courant du processeur *(return address et program status register)*, change son mode d'opération, recherche l'adresse de la routine pour le traitement de l'interruption dans la table des vecteurs d'interruptions, puis entre dans cette routine de traitement d'interruption
  - cette dernière sauve le contenu de tous les registres du µP
  - elle détermine l'identité du périphérique ayant levé l'interruption et le sert
  - elle restaure le contenu des registres du µP et se termine (retourne)
  - le µP restaure son registre d'état et poursuit l'exécution du programme où il a été interrompu

Contexte et latence

- L'action de sauver les registres du µP lorsqu'une interruption est levée, ou de les restaurer à la fin du traitement, s'appelle la commutation de contexte *(context switching)*
- Le temps qui s'écoule entre l'activation du signal d'interruption et l'exécution de la première instruction de la routine de traitement d'interruption est appelée latence d'interruption *(interrupt latency)*



Latence d'interruption

- La latence d'interruption dépend de 3 facteurs
  - Temps de désactivation des interruptions
    Temps durant lequel le µP n'est pas autorisé à traiter d'interruptions (interruptions désactivées p. ex. via les fanions du registre d'état)
  - Temps d'exécution de la commutation de contexte
    Temps utilisé par le µP pour sauver son contexte (return address et CPSR) et rechercher la routine de traitement d'interruption, ainsi que le temps utilisé par cette dernière pour la sauvegarde des registres du µP
  - Temps d'exécution de l'instruction machine la plus lente
    Temps utilisé par le µP pour terminer l'instruction en cours d'exécution lors de la levée d'un interruption
- Dans un système fréquemment interrompu, la valeur de cette latence influence considérablement les performances du système

Gigue d'interruption

- La variation de la latence d'interruption est appelée gigue d'interruption *(interrupt jitter)*
  - Son amplitude est principalement due aux variations de la durée de désactivation des interruptions.
  - La gigue peut poser des problèmes sérieux dans des systèmes temps réel ayant des contraintes exigeantes

Table des vecteurs d'interruptions

- Le type d'interruption ou d'exception détermine le mode du fonctionnement du µP
- Pour traiter une interruption, le µP exécute une instruction placée dans une table appelée table des vecteurs d'interruptions *(interrupt vector table)* correspondant au type d'interruption
- L'instruction située dans cette table permet au µP de charger dans son PC l'adresse de la routine de traitement de l'interruption et ainsi d'appeler la routine de traitement
- Chaque type d'interruption implémente sa propre routine de traitement

Table des vecteurs d'interruptions des µP ARM

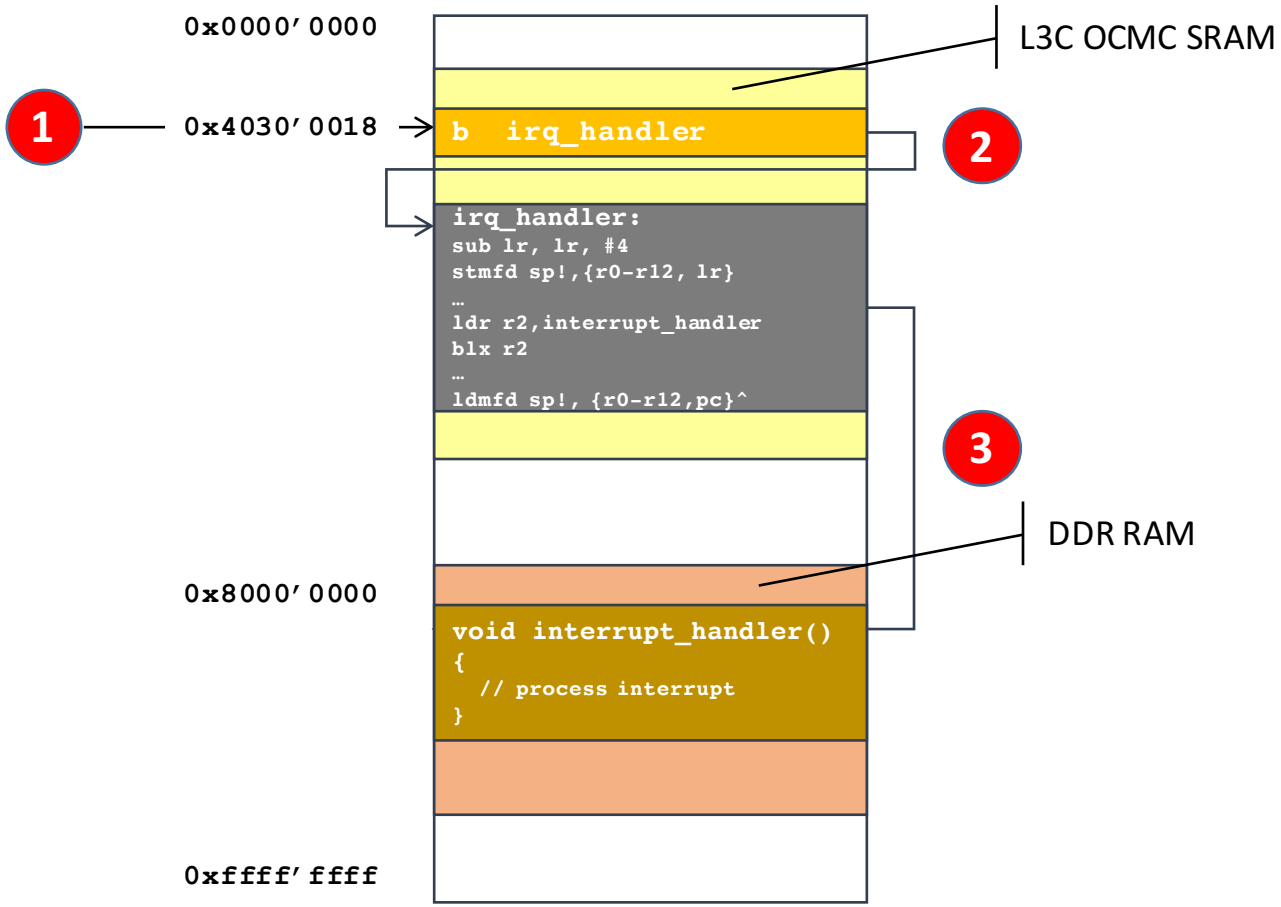
- Sur les µP ARM, cette table est généralement placée à l'adresse 0
- Sur certaines variantes, p. ex. le Cortex-A8, celle-ci peut être placée librement dans l'espace adressable du µP

Vecteur	Offset	Instruction	Mode
Reset	0x0000'0000	b reset_handler	Supervisor
Undefined instruction	0x0000'0004	b undef_handler	Undefined
Software Interrupt (SVC)	0x0000'0008	b svc_handler	Supervisor
Prefetch Abort (instruction fetch)	0x0000'000C	b inst_handler	Abort
Data Abort (data access)	0x0000'0010	b data_handler	Abort
Reserved	0x0000'0014	b reserved_handler	---
IRQ (interrupt)	0x0000'0018	b irq_handler	IRQ
FIQ (fast interrupt)	0x0000'001C	b fiq_handler	FIQ

- Le µP de TI AM3358 basé sur un Cortex-A8 permet de placer librement la table de vecteurs

TI AM3358 - Un exemple d'implémentation...

- Le µP TI AM3358 dispose de plusieurs mémoires SRAM internes
- La mémoire "L3 OCMC" située à l'adresse 0x4030'0000 d'une taille de 64KiB servira à stocker la table des vecteurs d'interruptions ainsi que la routine de traitement de bas niveau



Modes de fonctionnement du µP TI AM3358

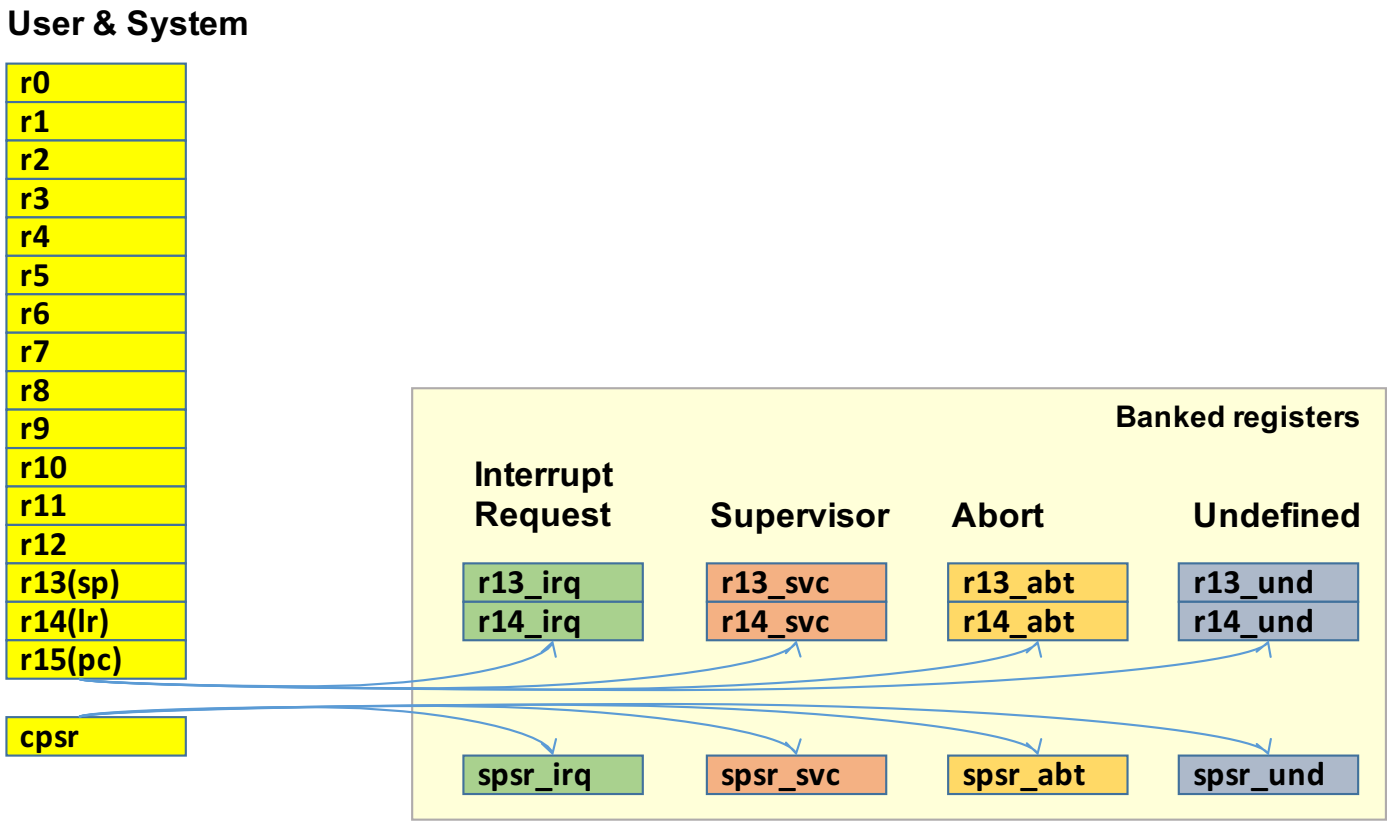
- Le µP TI AM3358 ne connaît que 6 modes de fonctionnement

Mode	Abrv.	Code	Description	Accès
User	usr	0x10	Mode d'exécution normal de programme par les utilisateurs (OS)	Limité
~~FIQ~~	~~fiq~~	~~0x11~~	~~Mode actif lors du traitement d'interruptions demandant un traitement rapide~~	~~Illimité~~
IRQ	irq	0x12	Mode actif lors du traitement d'interruptions normales	Illimité
Supervisor	**svc**	**0x13**	**Mode utilisé lors d'exécution de code propre au système d'exploitation (OS)**	**Illimité**
Abort	abt	0x17	Implémente les mécanisme de mémoire virtuelle	Illimité
Undefined	und	0x1b	Software émulation	Illimité
System	sys	0x1f	Tâches mode privilégié du système d'exploitation	Illimité

- Le mode FIQ n'est pas implémenté sur le µP TI AM3358
- Après reset, le µP fonctionne dans le module "Supervisor"

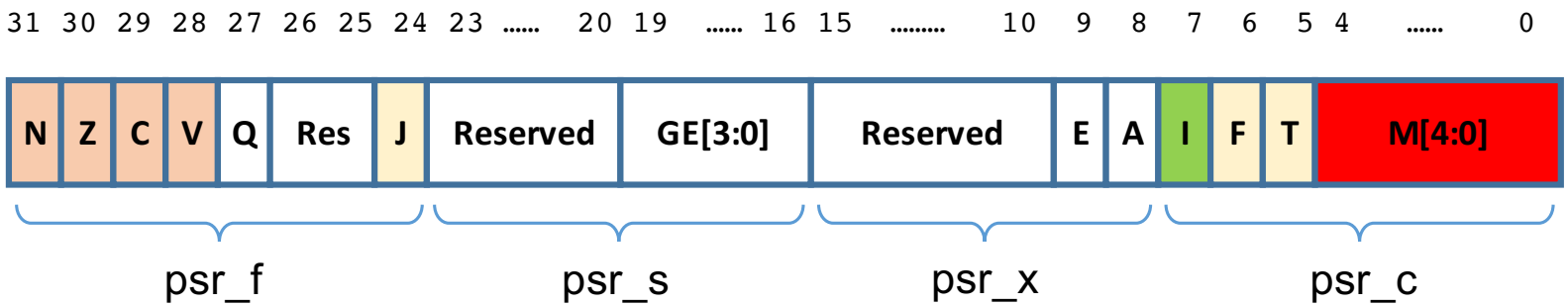
Sauvegarde des registres internes du µP

- Pour chaque mode de fonctionnement, le µP ARM duplique les registres SP, LR et SPSR
- Ces registres servent à sauver l'état du µP lors de la levée d'une interruption



Etat du µP

- Le registre d'état (PSR) reflète l'état du µP et indique son mode de fonctionnement (bits M[4 : 0])
- Ce registre est accessible depuis tous les modes, cependant en mode "User", seul les bits du champ "psr_f" sont modifiables



- Les bits M[4 : 0] déterminent le mode de fonctionnement du µP
- Les bits I et F sont les bits pour autoriser ou bloquer les interruptions.
  - Le bit I permet de bloquer les IRQ (interrupt request) s'il est mis à 1
  - Le bit F n'est pas disponible sur le µP TI AM3358.

Entrée dans la routine de traitement d'interruption

- Lorsqu'une interruption est levée, le µP effectue les opérations suivantes
  - Sauvegarde de l'adresse de retour et du CPSR dans les registre LR et SPSR du mode correspondant à l'interruption
    R14_<mode> = lien de retour
    SPSR_<mode> = CPSR
  - Mise à jour du mode d'opération du µP via le registre CPSR
    M = code correspond au mode d'opération
    I = 1 désactive les IRQ
  - Appel de la routine d'interruption
    PC = adresse du vecteur d'interruption

Sortie de la routine de traitement d'interruption

- Pour sortir de la routine de traitement, il faut restaurer l'état du CPSR ainsi que charger le PC avec le contenu du LR

```
subs  pc, lr, #offset // 's' permet de restaurer le contenu du CPSR
```

- L'instruction ci-dessus doit être utilisée pour corriger la valeur de retour avec un offset spécifié par ARM

Event	Offset	Return from handler
Reset	n/a	n/a
Data Abort	4	subs pc, lr, #4
IRQ	4	subs pc, lr, #4
Prefetch Abort	4	subs pc, lr, #4
SWI	0	movs pc, lr
Undefined Instruction	0	movs pc, lr

Sauvegarde et restauration du contexte

- Une technique courante pour sauver l'état du µP consiste à corriger l'adresse de retour à l'entrée de la routine d'interruption et de la sauver avec les autres registres
- Sauvegarde des registres à l'entrée

```
    sub    lr, #4              // ajustement de l'adresse de retour
    stmfd  sp!, {r0-r12,lr}    // sauvegarde des registres sur la pile
```

- Appel de la routine spécifique au traitement de l'interruption
- Restauration des registres et du contexte du µP à la sortie

```
    ldmfd  sp!, {r0-r12,pc}^   // restaure le contexte du µP et retourne
```

  - ^ permet de restaurer le CPSR depuis le SPSR
  - PC remplace LR afin de retourner directement au programme interrompu

Pointeurs de piles

- Chaque mode d'opération du µP possède sa propre pile, laquelle est accessible via le registre R13/SP du mode correspondant
- Pour changer le contenu du registre SP, il faut changer le mode d'opération du µP vers le mode souhaité et ensuite il est possible changer le contenu du registre SP

```
// initialise IRQ stack pointer
msr   cpsr_c, #0xd2 // switch to IRQ mode
ldr   sp, =irq_stack_top
```