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

Systèmes Embarqués 1 & 2

p.01 - Interruptions

Classes T-2/I-2 // 2018-2019

Daniel Gachet | HEIA-FR/TIC

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz

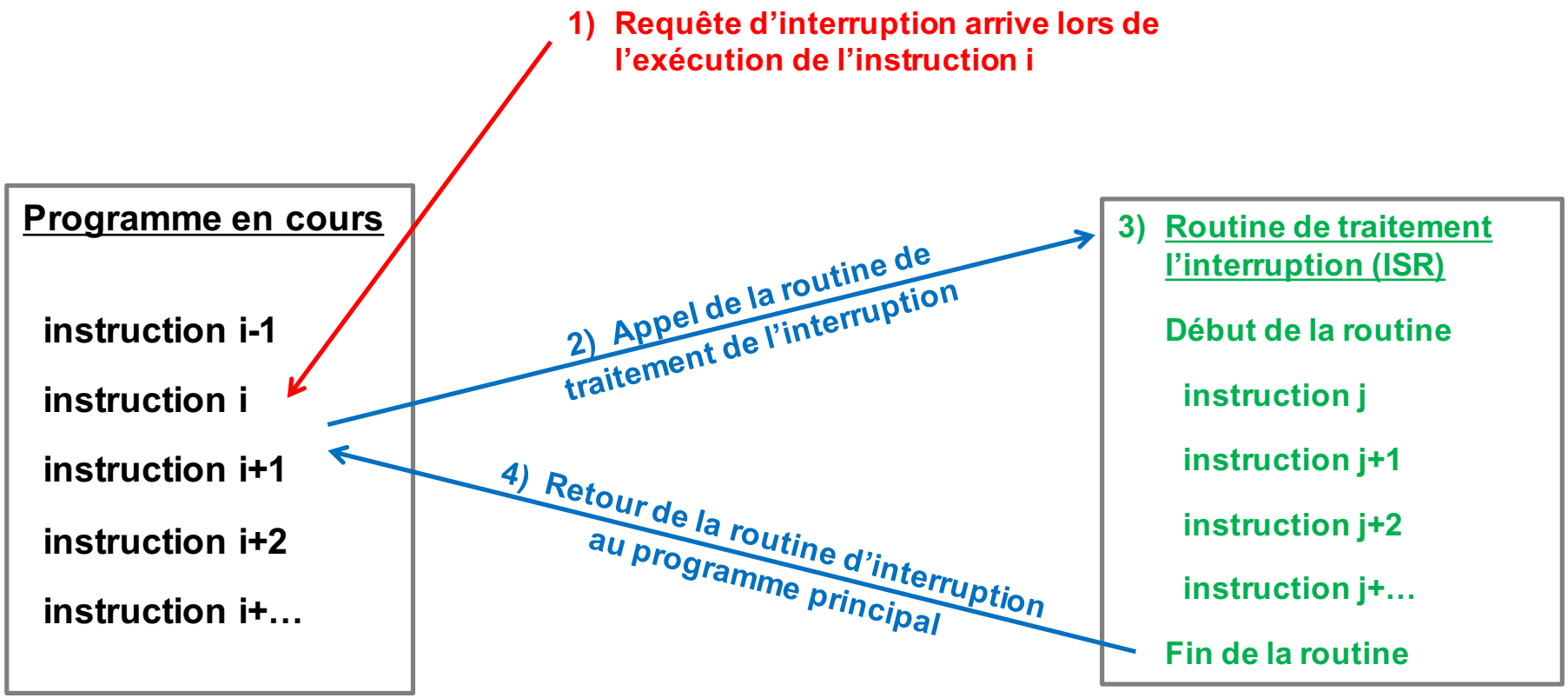
Contenu

- 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
- Principe des interruptions matérielles
- Interruptions multiples
- Système d'interruptions des µP ARM Cortex-A8 et TI AM335x

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)

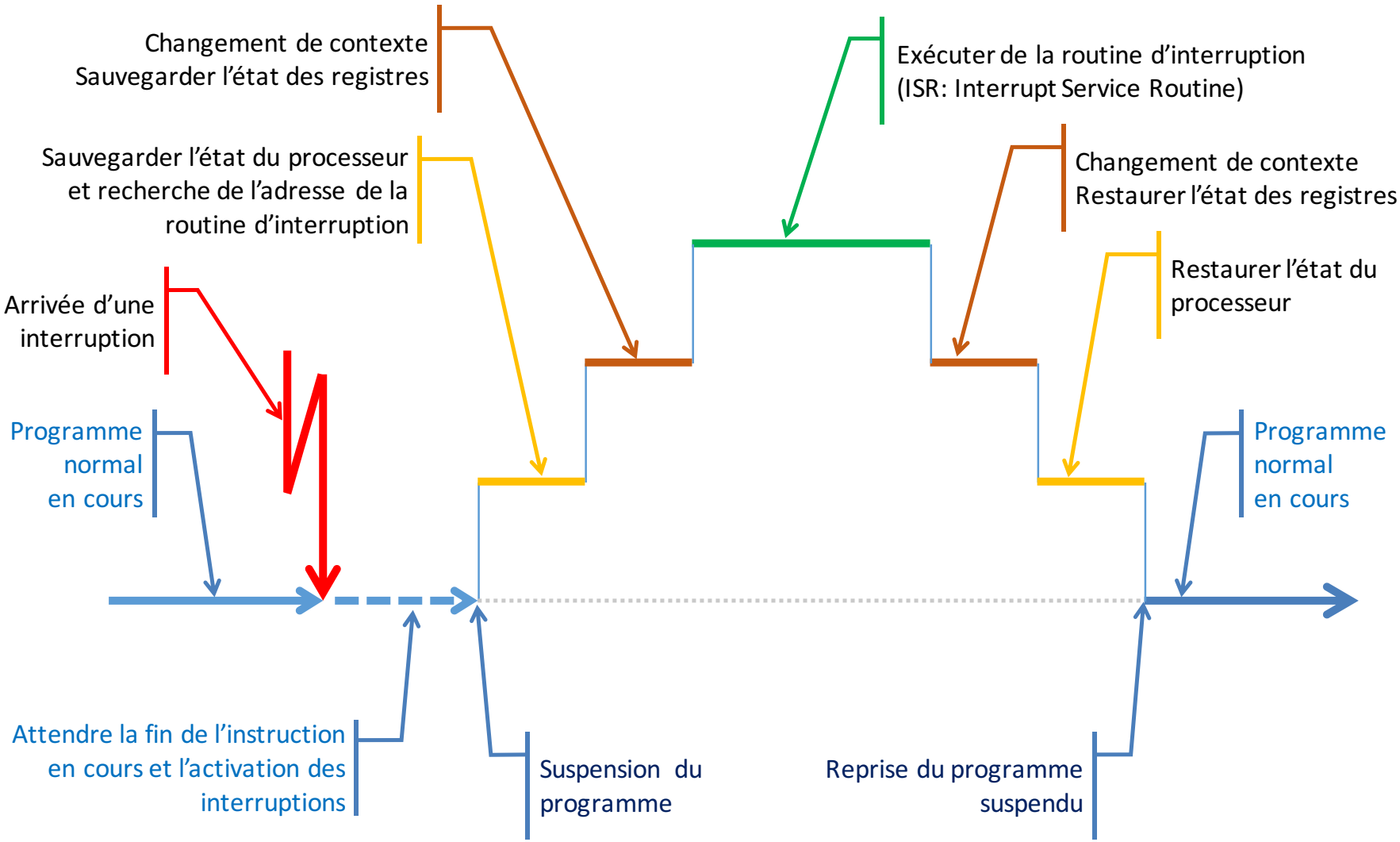
Interruptions matérielles

- 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

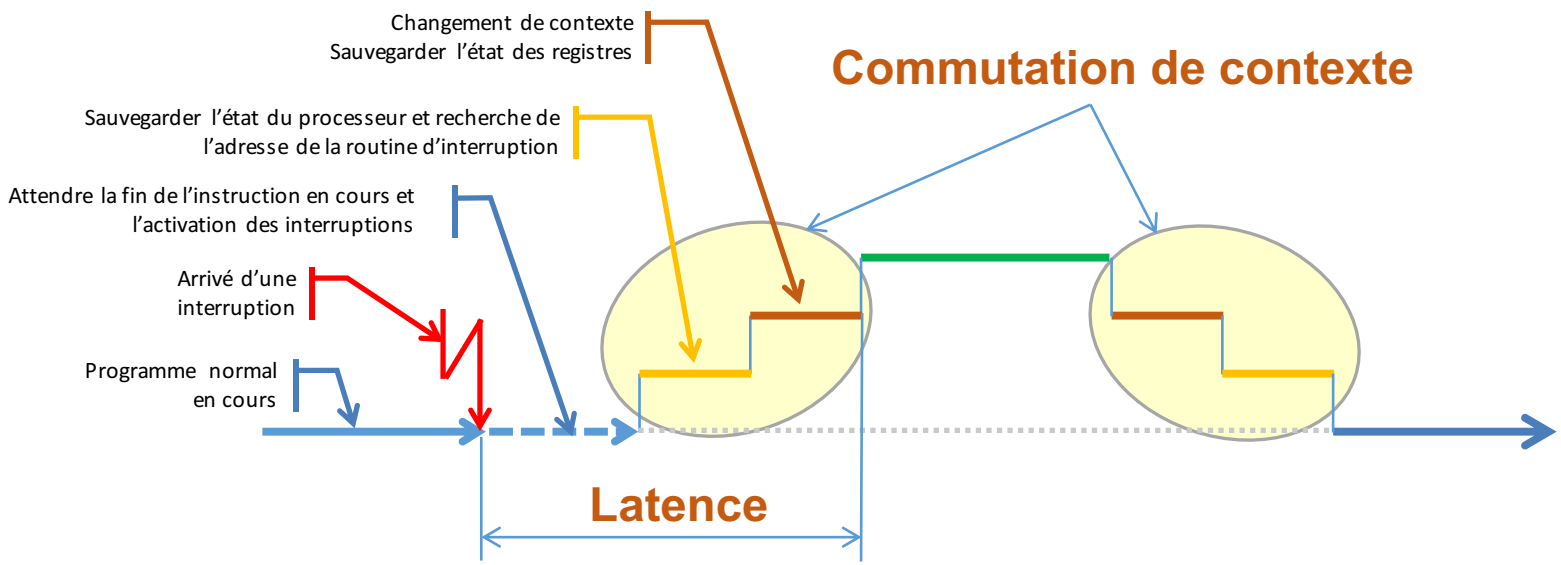


Séquence d'interruption (II)

- 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 est due à un facteur principal
  - 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)
- 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

Modes de fonctionnement du µP TI AM3358

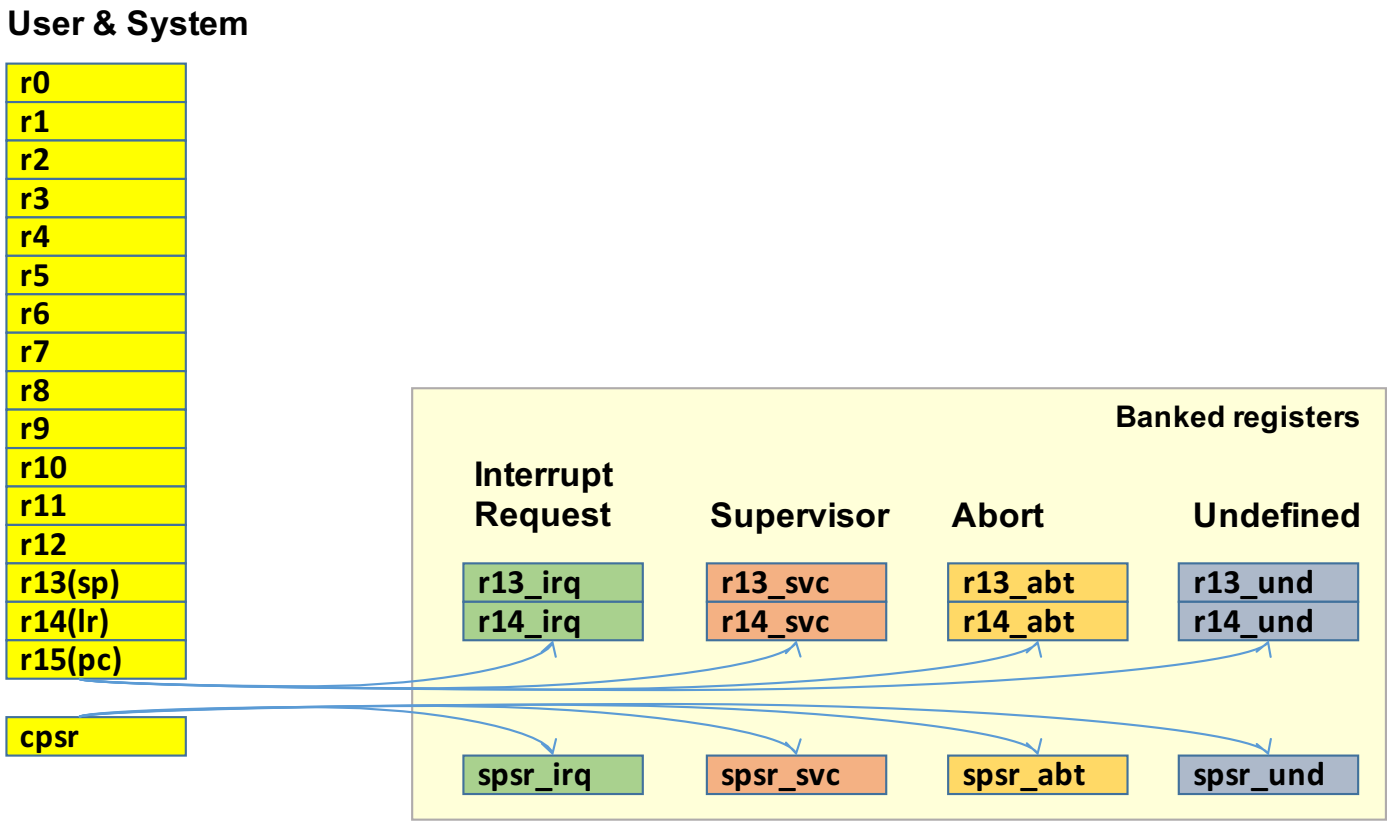
- Le µP TI AM3358 ne connaît que 6 modes de fonctionnement

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"

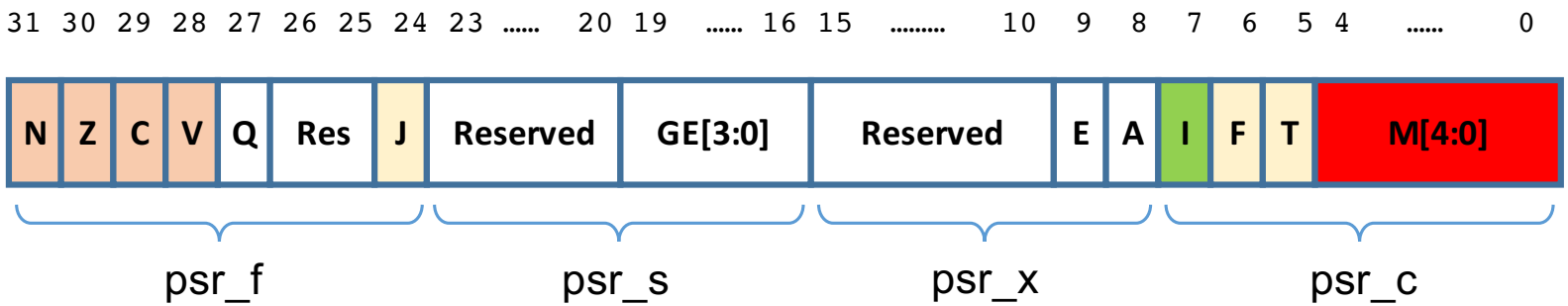
Sauvegarde des registres internes du µP

- 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

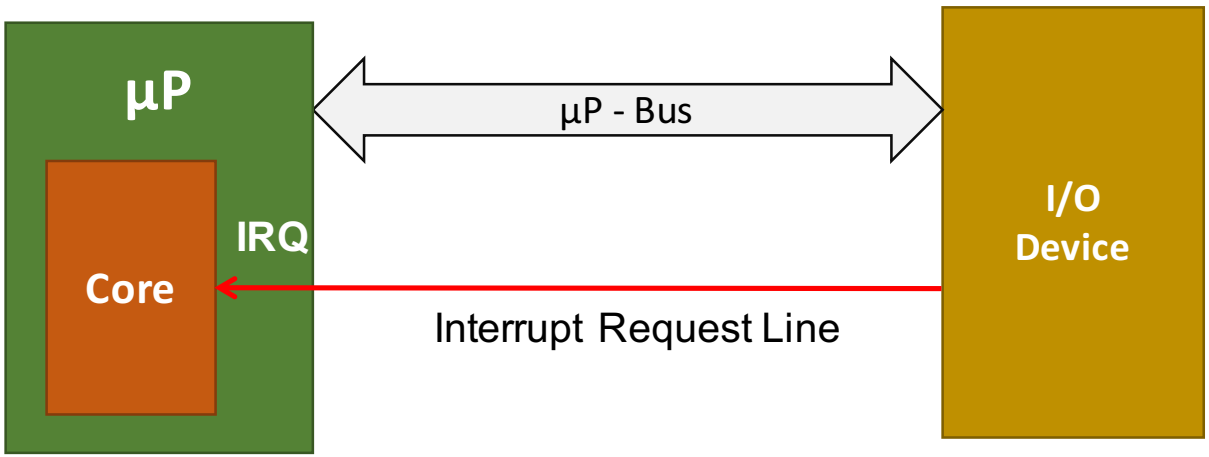
Pointeurs de piles

- 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
```

Interruptions matérielles - Le principe

- Le périphérique signale sa requête d'interruption au µP en activant la ligne de requêtes d'interruptions *(Interrupt Request Line)*



- Lorsque le µP reçoit la requête
  - il termine l'exécution de l'instruction en cours
  - il sauve l'état courant (état de ses registres, cpsr et pc)
  - il exécute une routine de traitement d'interruption *(Interrupt Handler)* pour servir le périphérique
  - il retourne au programme interrompu après avoir servi le périphérique

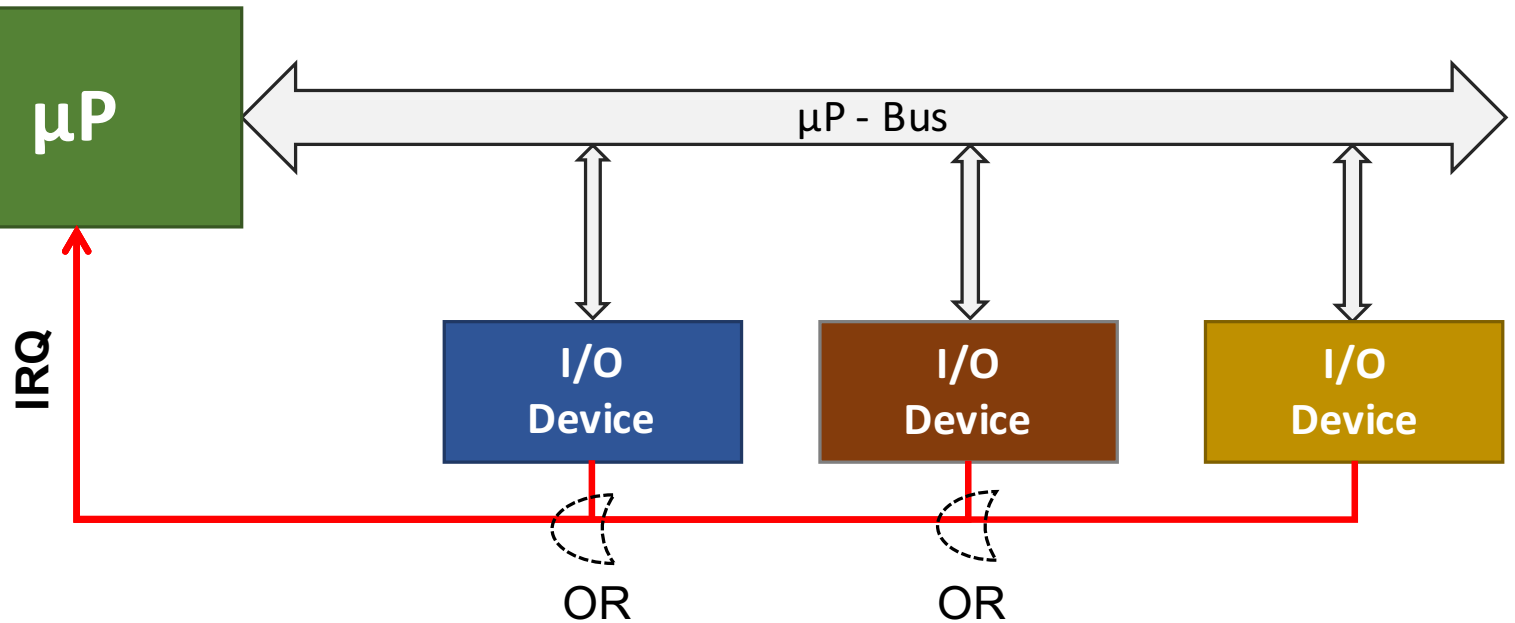
- Identifier le périphérique
  - Lorsque plusieurs périphériques sont connectés sur le même µP, le traitement des interruptions suppose que le µP puisse identifier le périphérique ayant levé l'interruption
- Arbitrer les requêtes multiples
  - Si plusieurs périphériques peuvent lancer simultanément une requête d'interruption, le µP doit être capable d'arbitrer les conflits d'accès en servant l'un après l'autre les périphériques demandeurs, ceci en fonction du degré d'urgence des besoins à satisfaire

Interruptions matérielles - Les techniques

- Différentes techniques permettent au µP de déterminer la source de l'interruption
  - Scrutation logicielle
  - Priorité d'interruption
  - Interruption vectorisée

Scrutation logicielle

- Une seule ligne de requêtes d'interruptions pour tous les périphériques



Scrutation logicielle (II)

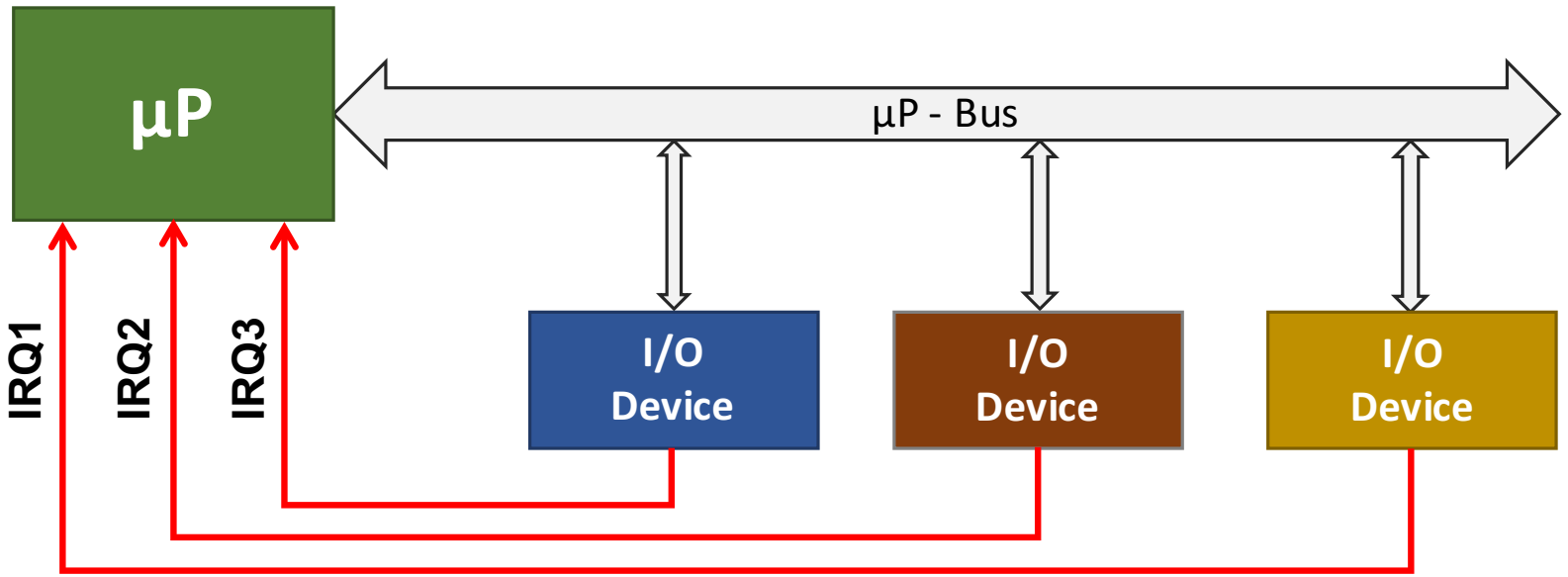
- Pour la scrutation logicielle if suffit d'une ligne unique de requêtes d'interruptions sur laquelle tous les périphériques sont branchés en parallèle pour former un OU câblé
- Les périphériques sont équipés de registres de contrôle et d'état
  - Ces registres permettent de gérer le périphérique ansi que de bloquer ou d'autoriser la levée d'interruptions par le périphérique
  - Ils permettent également d'identifier le périphérique ayant levé l'interruption
- Si une interruption apparaît, le µP suspend le programme en cours et exécute la routine de traitement d'interruptions
- Cette routine interroge successivement tous les périphériques pour déterminer celui qui a activé le signal d'interruptions

Scrutation logicielle (III)

- Deux méthodes de scrutation logicielle *(interrupt polling)*
  - Priorité fixe
    les périphériques sont scrutés dans un ordre précis et déterminé à l'avance *(fixed priority)*. Le 1er périphérique rencontré ayant levé une interruption est servi, puis le 2^{e} et ainsi de suite...
  - Tourniquet
    les périphériqes sont ordonnés comme dans la première méthode, mais la recherche débute depuis le dernier périphérique servi *(round-robin)*. La recherche recommence au début, lorsque la fin de la liste est atteinte.
- La scrutation logicielle est simple à mettre en oeuvre, mais peut prendre énormément de temps selon le nombre de périphériques connectés au µP

Priorité d'interruption

- Une ligne de requêtes d'interruptions dédiée à chaque périphérique

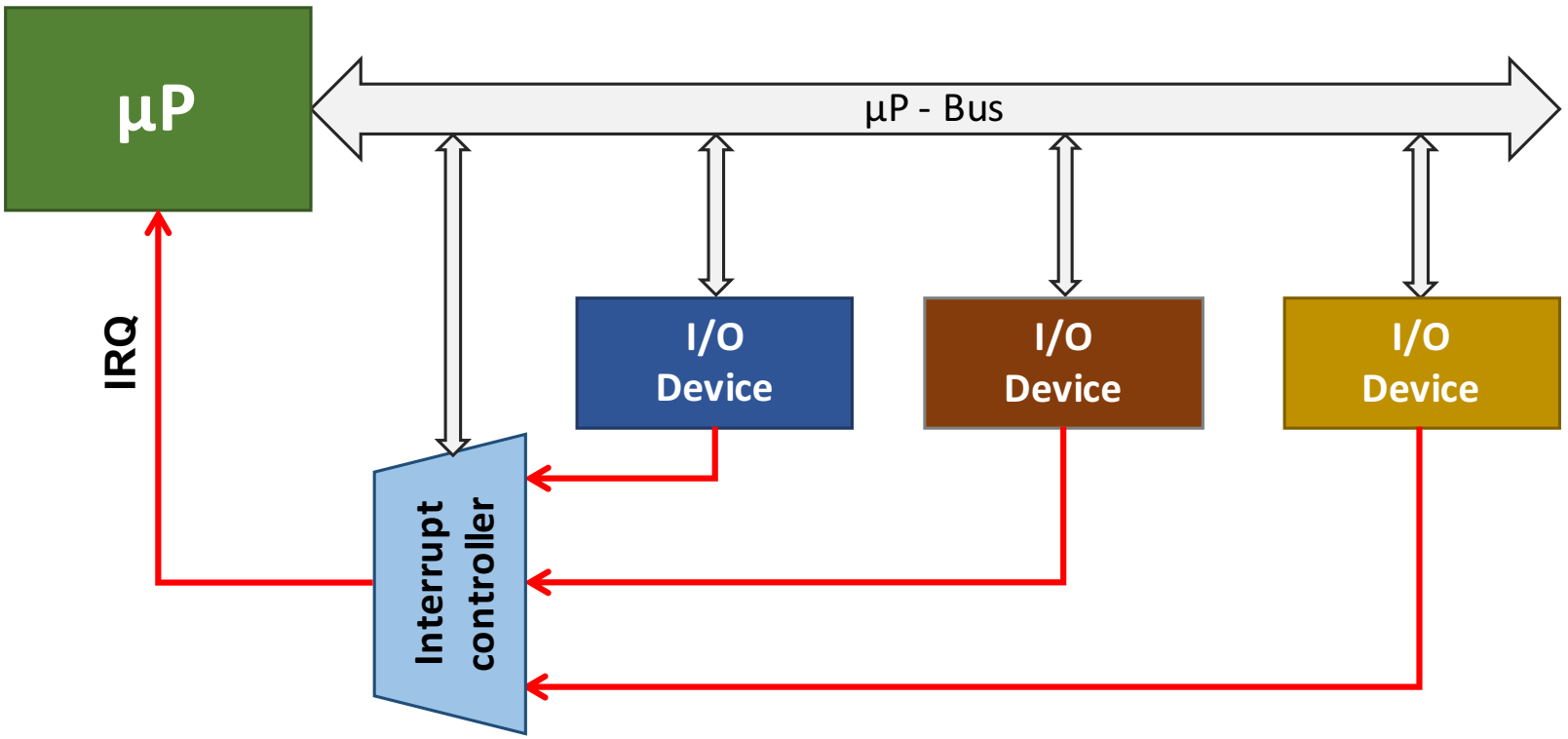


Priorité d'interruption (II)

- Afin d'identifier et de prioriser les interruptions, les µP mettent à disposition plusieurs lignes de requêtes d'interruptions
- Le µP affecte à chacune des lignes de requêtes d'interruptions une priorité matérielle fixe et unique, par exemple aux lignes IRQ1, IRQ2, IRQ3, le µP donnera les priorités 1, 2 et 3
- Chaque périphérique dispose de sa propre ligne de requêtes d'interruptions
- Lorsqu'une ou plusieurs requêtes d'interruptions apparaissent, le µP examine les lignes de requêtes d'interruptions pour déterminer la ligne active la plus prioritaire afin de servir le périphérique correspondant

Interruption vectorisée

- Un contrôleur capable de connecter, de prioriser et d'identifier plusieurs périphériques

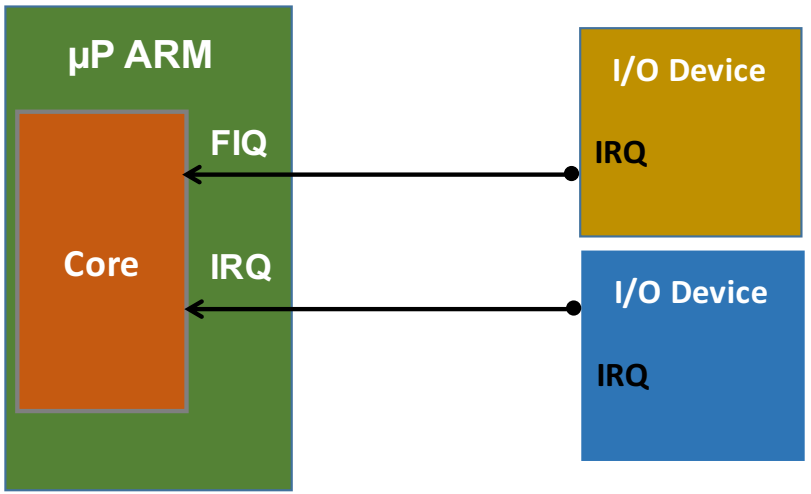


Interruption vectorisée (II)

- Une interruption vectorisée est définie par la capacité d'un contrôleur d'interruptions de connecter, d'identifier et prioriser une grande quantité de périphériques
- Chaque périphérique est connecté directement au contrôleur par une ligne de requêtes d'interruptions unique
- Le contrôleur dispose d'une logique interne pour prioriser et identifier les différentes sources d'interruptions
- Cette logique lui permet d'activer la ligne de requêtes d'interruptions du µP si un ou plusieurs périphériques ont signalé une interruption
- Lorsqu'une interruption est levée, la routine de traitement appelée par le µP obtient le vecteur d'interruption (l'identité du périphérique) en lisant un registre du contrôleur d'interruptions

ARM – Système d'interruption

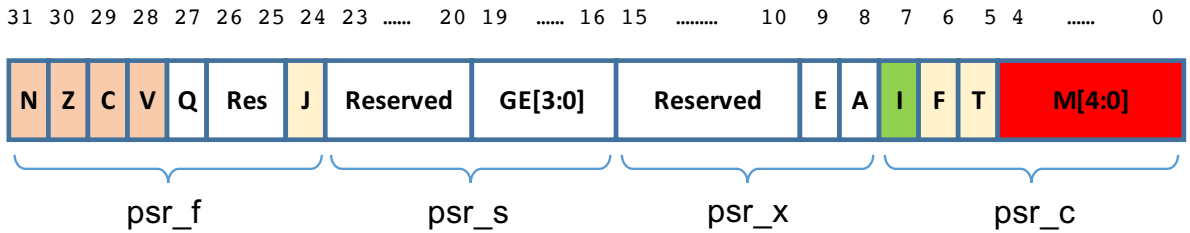
- Les µP ARM ne disposent que de deux lignes de requêtes d'interruptions, la Interrupt Request (IRQ) et la Fast Interrupt Request (FIQ)



- Ces deux lignes permettent de connecter très simplement des périphériques au µP ARM
- Chacune de ces lignes correspond à un niveau de priorité fixe et prédéfini

ARM – Acceptation des interruptions

- Les bits I et F du registre CPSR offrent au µP la possibilité d'autoriser ou de bloquer le traitement des interruptions matérielles



- Le bit I permet d'autoriser les interruptions provenant de la ligne d'interruptions IRQ (I==0) et le bit F de la ligne FIQ (F == 0)
- Lorsqu'une interruption est levée, le µP change de mode et sauve son état dans les registres LR et SPSR, puis il bloque le traitement d'interruptions futures en manipulant les bits I et F
  - Si une IRQ est levée, alors I est mis à 1 et F reste inchangé
  - Si une FIQ est levée, alors I et F sont mis à 1
- Finalement, le µP poursuit le traitement de l'interruption en appelant la routine de traitement placée dans sa table de vecteurs

ARM - Priorité des interruptions/exceptions

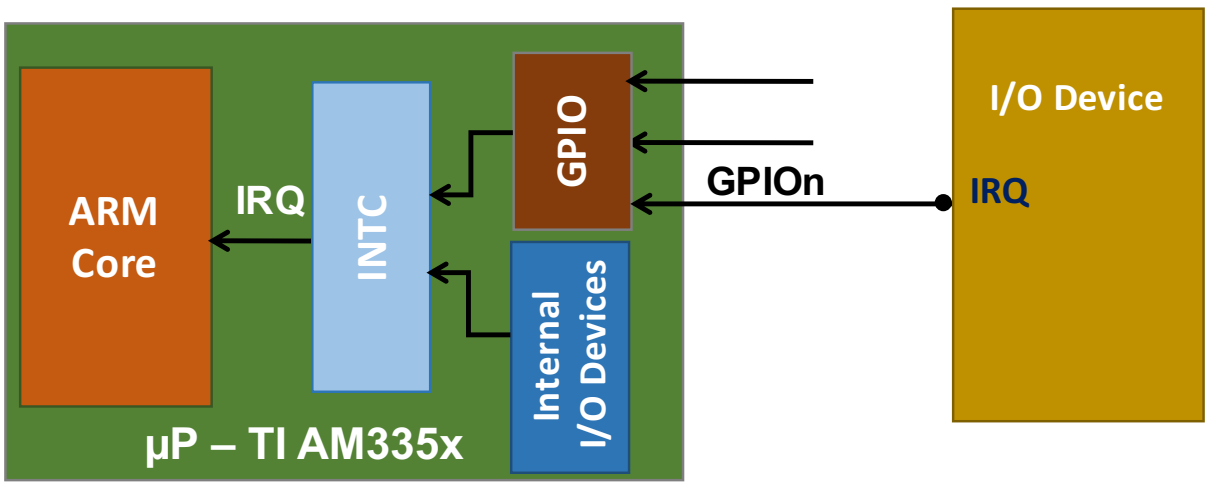
- La notion de priorité des interruptions/exceptions ne prend tout son sens que lorsque celles-ci surviennent simultanément
- Les µP ARM classe les exceptions sur 7 niveaux, le niveau 1 le plus prioritaire et le niveau 7 le moins

Priority		Exception
Highest	1	Reset
	2	Data Abort
	3	FIQ
	4	IRQ
	5	Imprecise Abort
	6	Prefetch Abort
Lowest	7	Undefined instruction / SVC / BKPT

- Le µP commence le traitement des interruptions par la plus prioritaire

TI AM335x - Système d'interruptions

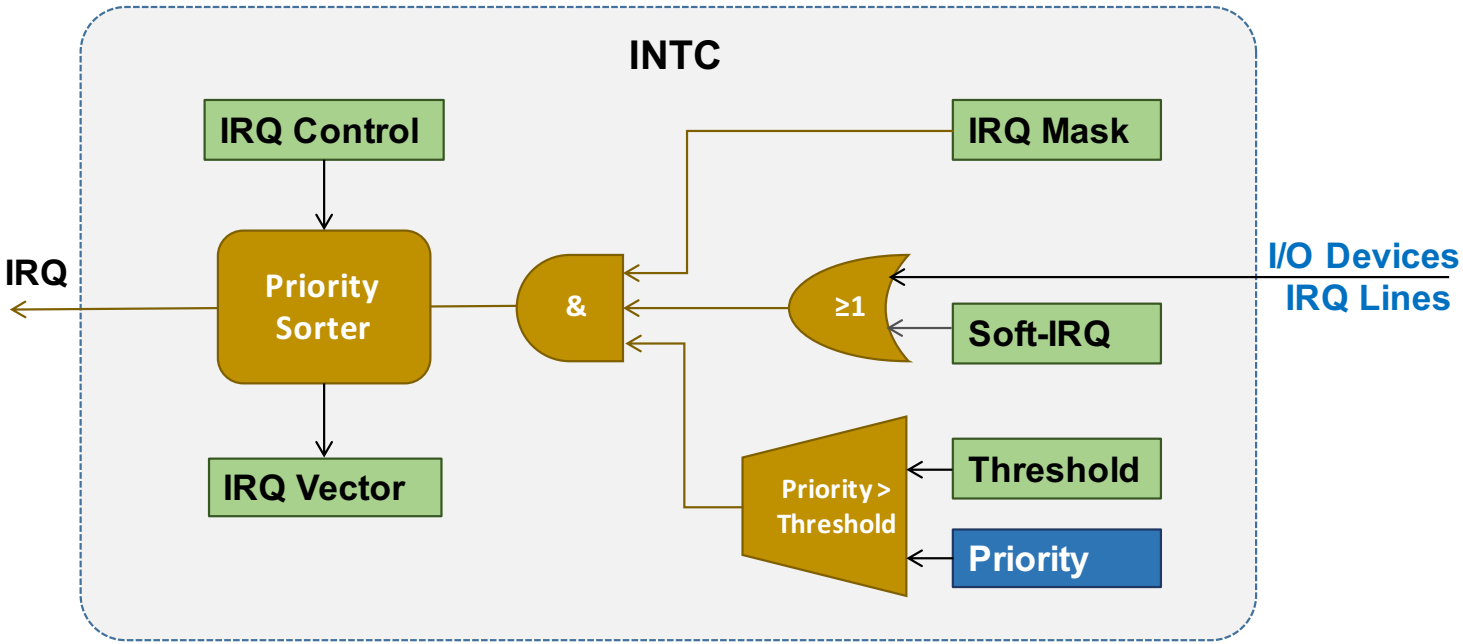
- Pour permettre d'interfacer avec un plus grand nombre de périphériques, le µP ARM TI AM335x dispose d'un contrôleur d'interruptions (INTC)



- Ce contrôleur est connecté au µP que par la ligne IRQ (pas de FIQ)
- L'INTC permet de collecter les requêtes d'interruptions de 128 sources différentes (périphériques internes au µP)
- Il est capable de prioriser ces sources et de générer un vecteur d'interruption pour la source la plus prioritaire

TI AM335x - INTC - Block Diagram

- Le contrôleur INTC permet de prioriser et de bloquer chaque ligne de requêtes d'interruptions, ainsi que de simuler par logiciel la levée d'une interruption



- Lorsque des interruptions sont levées, le contrôleur sélectionne, grâce à au niveau de priorité attribué aux sources d'interruptions, la source la plus prioritaire et la sert en premier

TI AM335x - INTC - Niveaux de priorité

- Le contrôleur INTC implémente un décodeur de priorités ne prenant en compte que les sources ayant un niveau suffisant pour être traitées
- Si le niveau de priorité de la source d'interruption est inférieur ou égale à une valeur prédéfinie, le threshold, la requête sera bloquée jusqu'à ce que le threshold soit ajusté

Threshold	Source Priority Level						
	0	1	2	3	...	126	127
0xff	√	√	√	√	√	√	√
0	x	√	√	√	√	√	√
1	x	x	√	√	√	√	√
2	x	x	x	√	√	√	√
3	x	x	x	x	√	√	√
...	x	x	x	x	x	√	√
126	x	x	x	x	x	x	√
127	x	x	x	x	x	x	x

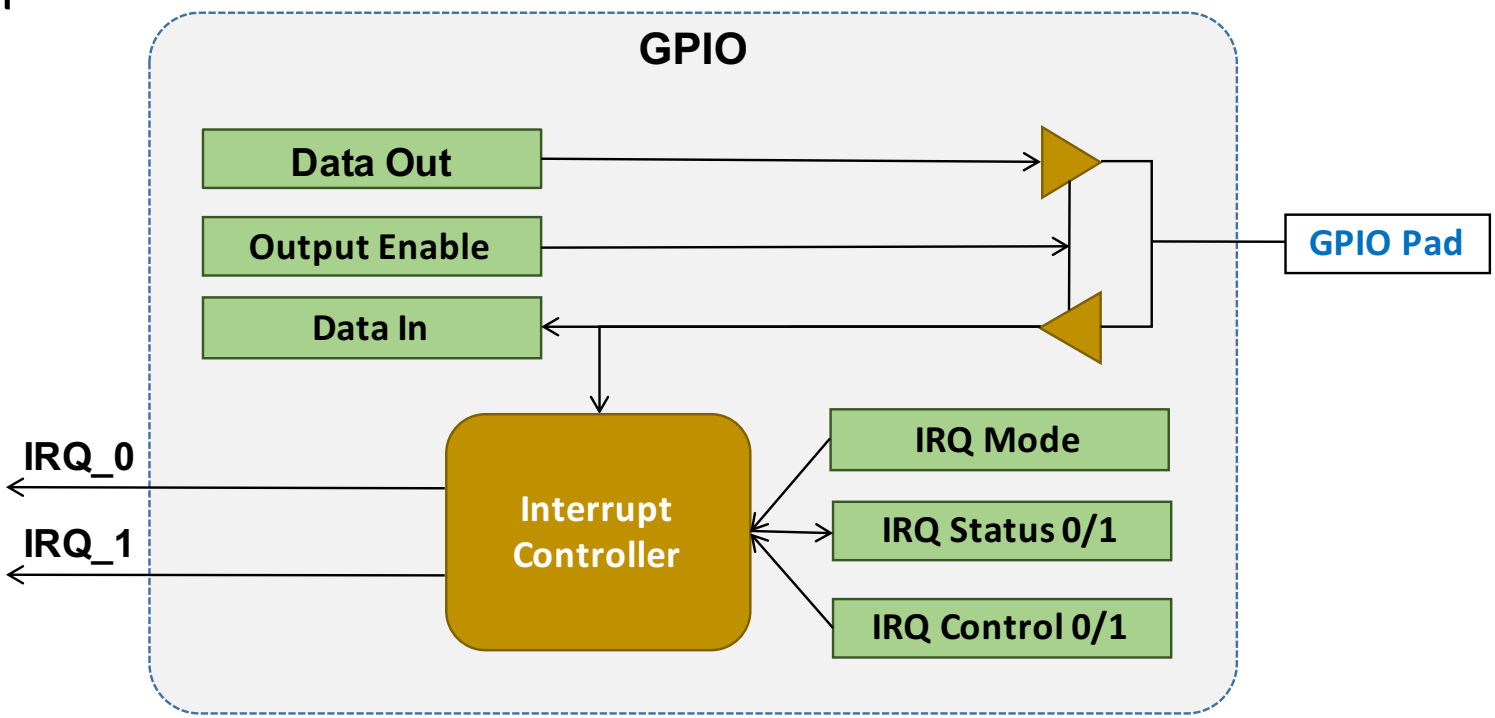
Current Threshold →

Unaccepted interrupts

Accepted interrupts

TI AM335x - GPIO - Block Diagram

- Le µP TI AM335x compte 4 ports GPIO (port 0 à port 3) avec chacun 32 broches (gpio-pads)
- Chaque broche peut être configurée en entrée ou en sortie
- Si la broche est configurée en entrée, elle peut également servir de ligne de requêtes d'interruptions pour des périphériques externes au µP



TI AM335x - GPIO - Configuration des interruptions

- Le module GPIO peut être configuré pour générer une interruption s'il détecte
  - un niveau haut du signal (high level)
  - un niveau bas du signal (low level)
  - un flanc descendant (falling edge)
  - un flanc montant (rising edge)

- Le module GPIO permet également de choisir pour chaque port si l'interruption à générer doit être sur la ligne 0 ou 1 (IRQ0 ou IRQ1)