

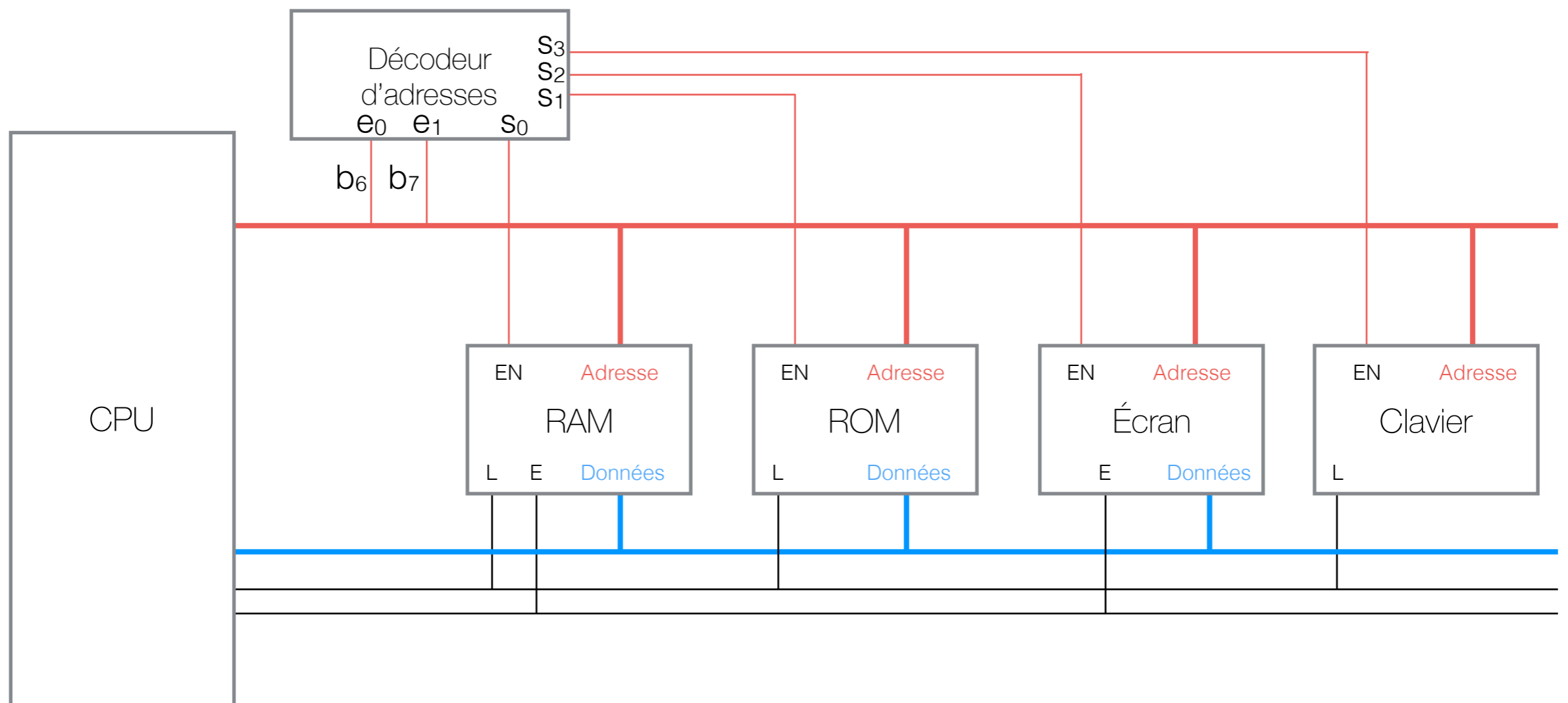
Les interruptions



GIF-1001 Ordinateurs: Structure et Applications
Jean-François Lalonde

Scénario 1

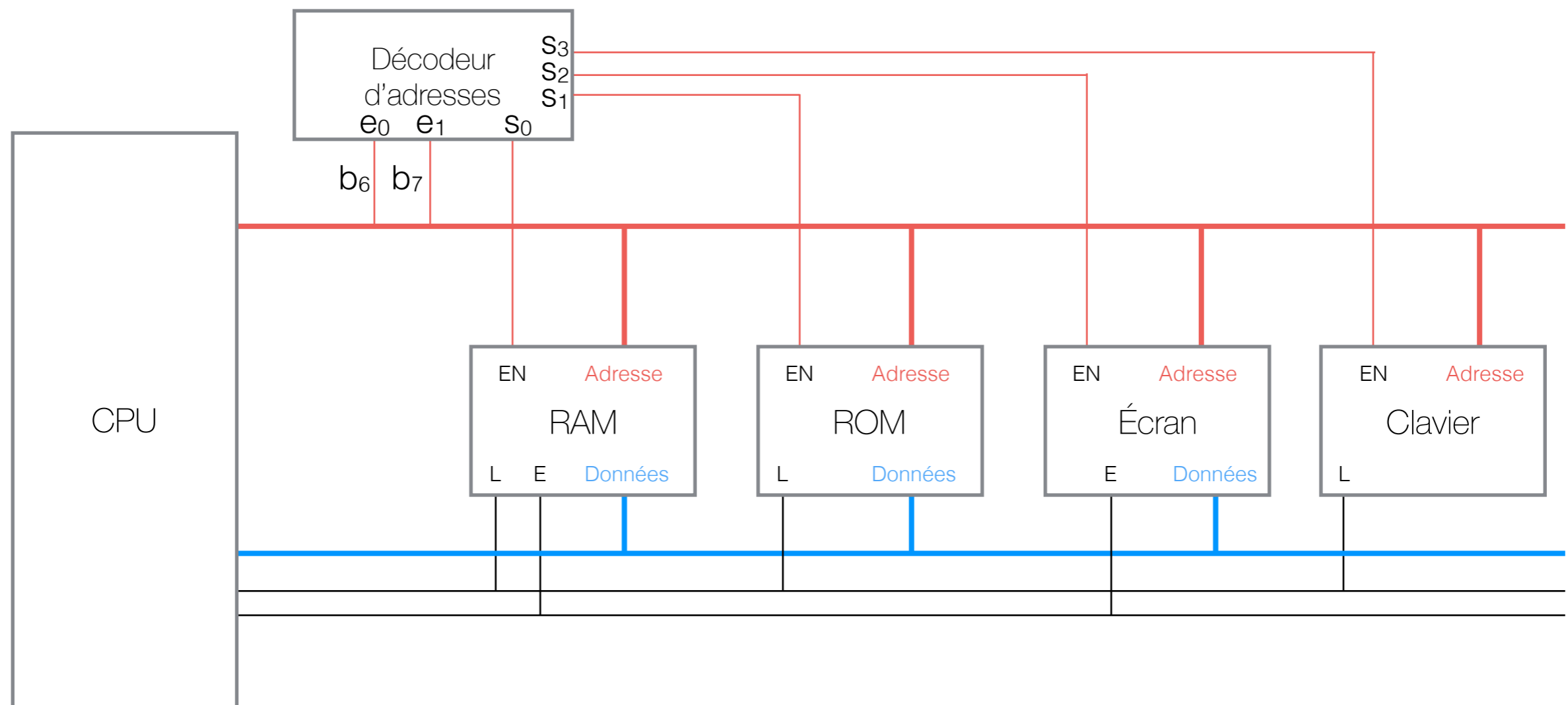
Un micro-processeur veut lire les valeurs écrites au clavier par un utilisateur.



Scénario 1

Un micro-processeur veut lire les valeurs écrites au clavier par un utilisateur.

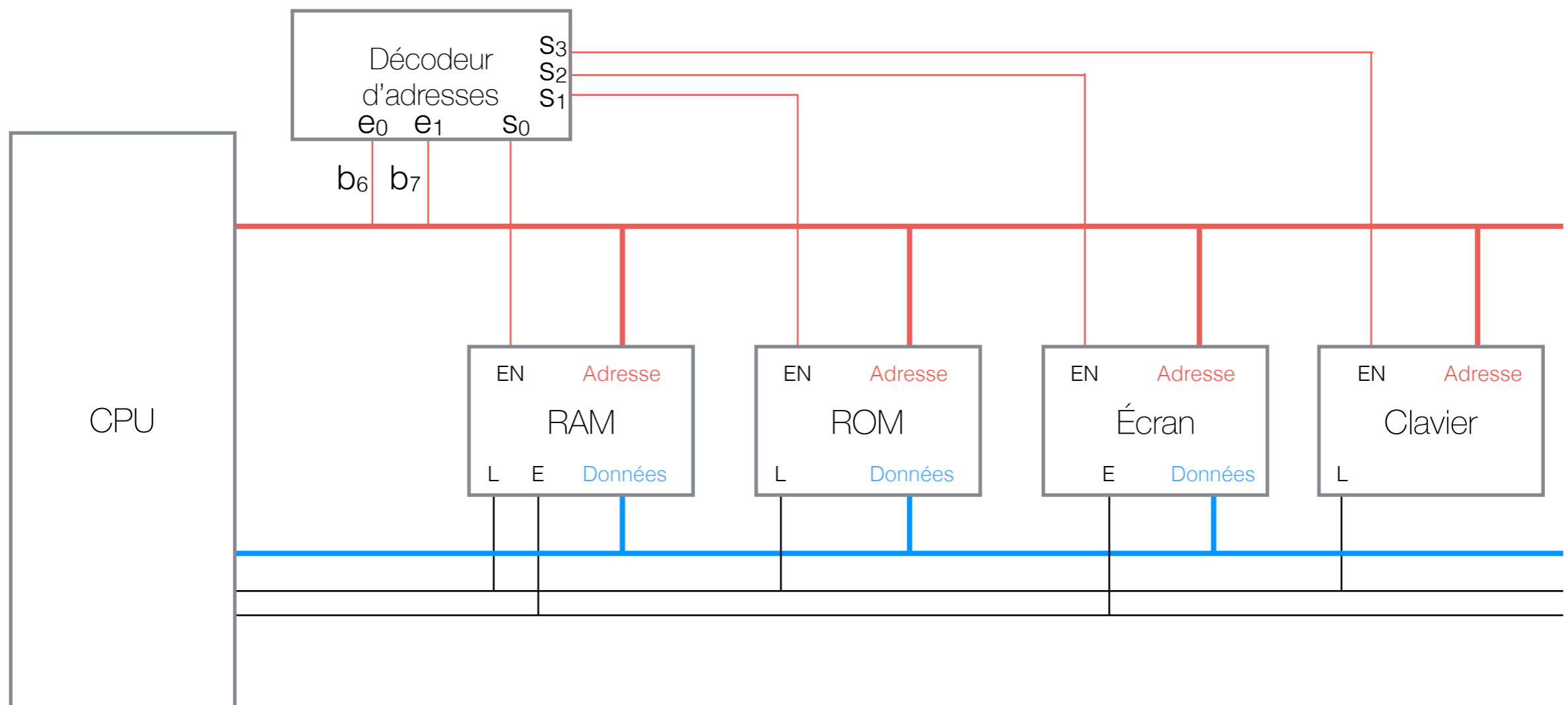
Option #1: interroger le clavier périodiquement pour voir s'il a reçu une nouvelle valeur. Sinon, attendre.



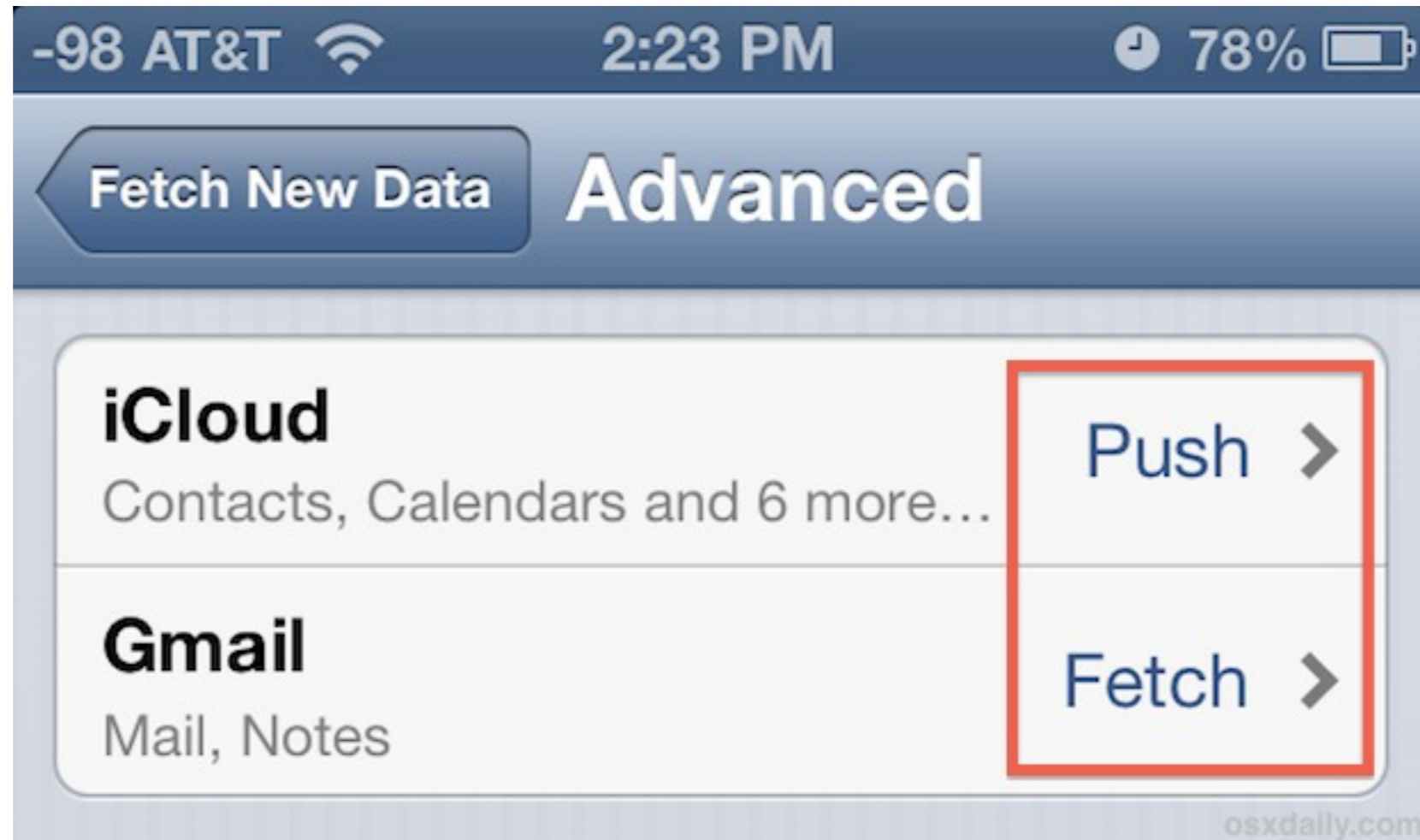
Scénario 1

Un micro-processeur veut lire les valeurs écrites au clavier par un utilisateur.

Option #2: c'est le clavier qui «interrompt» le micro-processeur pour l'aviser qu'il a reçu une nouvelle valeur.



Scénario 2



Scénario 3



Mise en situation









https://www.youtube.com/watch?v=3B8jcKx8H_w

Traitement des interruptions

- Une interruption interrompt l'exécution des instructions par le microprocesseur.
- Lors d'une interruption:
 1. l'exécution du programme principal est suspendue;
 2. une routine (fonction) traitant l'interruption est exécutée;
 3. puis le programme principal est continué.
- Quelle est la différence entre une interruption et un branchement?
 - les interruptions peuvent survenir n'importe quand pendant l'exécution.

Traitement des interruptions

Interruption!

1. Terminer l'instruction en cours
2. Déterminer s'il faut traiter l'interruption.
3. Sauvegarder le contexte
4. Déterminer l'adresse de la routine de traitement de l'interruption
5. Exécuter cette routine

Traitement de l'interruption...

1. Restaurer le contexte
2. Reprendre là où le processeur était rendu

Types d'interruptions

- **Systeme**: reset, faute matérielle générale, etc.
- **Exception**: erreur au cours de l'exécution des instructions, par exemple:
 - instruction invalide
 - division par 0
 - tentative d'accès à de la mémoire protégée
- **Matérielles**: générées par les périphériques, par exemple:
 - le clavier dans notre scénario 1 plus haut
 - l'imprimante lorsqu'elle a terminé, etc.
- **Logicielles**: générées par le programmeur avec une instruction spéciale (présente dans tous les jeux d'instructions)

Les interruptions en ARM

- Il y a 7 types d'interruptions dans le microprocesseur ARM:
 1. **Reset**: redémarrage du microprocesseur
 2. **Interruption logicielle**: activée par l'instruction SVC pour utiliser les fonctions systèmes (par exemple pour communiquer avec les périphériques);
 3. **Data abort**: accès mémoire invalide lors d'un LDR;
 4. **Prefetch abort**: accès mémoire invalide lors de la lecture d'une instruction (étape « lecture » dans le pipeline)
 5. **Instruction indéfinie**: erreur dans le décodage d'une instruction
 6. **IRQ**: activée par un périphérique (**I**nterrupt **ReQ**uest)
 7. **FIQ**: activée par un périphérique, mais qui peut être traité plus rapidement qu'une IRQ (**F**ast **I**nterrupt **reQ**uest)

Priorités

- Les interruptions ont des priorités: une interruption de haute priorité peut interrompre une interruption ayant un niveau de priorité plus bas.
- Certaines interruptions peuvent survenir n'importe quand, même pendant une autre interruption.
- Certaines interruptions, comme «reset», ont une priorité (maximale pour «reset») qui ne peut pas être changée.

Priorités

Priorité	Interruption
Plus haute	Reset
	« Data abort »
	FIQ
	IRQ
	« Prefetch abort »
Plus basse	Interruption logicielle et instruction indéfinie (ne peuvent pas survenir en même temps... pourquoi?)

Interruptions imbriquées

- Qu'arrive-t-il si une interruption survient lorsqu'on traite une interruption?
- Cela dépend de la priorité
 - Si la priorité de la nouvelle interruption est plus élevée:
 - On interrompt l'exécution et on traite cette nouvelle interruption
 - Si la priorité de la nouvelle interruption est moins élevée:
 - On attend que le traitement de l'interruption à plus haute priorité soit terminé, et on traite cette nouvelle interruption par la suite

Traitement des interruptions

Interruption!

1. Terminer l'instruction en cours
2. Déterminer s'il faut traiter l'interruption.
3. Sauvegarder le contexte
4. Déterminer l'adresse de la routine de traitement de l'interruption
5. Exécuter cette routine

Traitement de l'interruption...

1. Restaurer le contexte
2. Reprendre là où le processeur était rendu

Traitement des interruptions

Interruption!

1. Terminer l'instruction en cours
2. Déterminer s'il faut traiter l'interruption.
3. Sauvegarder le contexte
- 4. Déterminer l'adresse de la routine de traitement de l'interruption**
5. Exécuter cette routine

Traitement de l'interruption...

1. Restaurer le contexte
2. Reprendre là où le processeur était rendu

Routines de traitement d'interruptions

- Les routines de traitement d'interruptions sont des fonctions « spéciales » que l'on appelle que pour traiter les interruptions
- Où sont-elles situées?
 - en mémoire!
- Comment fait-on pour savoir:
 - quelle routine exécuter pour quelle interruption?
 - à quelle adresse est cette routine?
- Grâce à la table des vecteurs d'interruption, pardi!

Table des vecteurs d'interruption

- Chaque entrée de la table contient une instruction qui « branche » vers la routine correspondante

Adresse	Interruption	Signification
0x00	Reset	redémarrage
0x04	Instruction indéfinie	problème lors du décodage
0x08	Interruption logicielle	demandée par le programmeur: instruction SVC
0x0C	« Prefetch abort »	« fetch » invalide
0x10	« Data abort »	accès mémoire invalide
0x14	Espace réservé	ne rien mettre ici
0x18	IRQ	« Interrupt ReQuest »: interruption matérielle générale
0x1C	FIQ	« Fast Interrupt reQuest »: interruption matérielle rapide

La TVI dans le code

```
SECTION INTVEC
```

```
B main
```

```
SECTION CODE
```

```
main
```

```
; Programme principal
```

```
LDR R0, b
```

```
LDR R1, c
```

```
ADD R1, R0, R1
```

```
STR R1, a
```

```
B main
```

```
SECTION DATA
```

```
a ALLOC32 1
```

```
b ASSIGN32 1
```

```
c ASSIGN32 2
```

Table des vecteurs d'interruption

Ici, nous ne supportons qu'une seule interruption,
laquelle?

Code principal

Données en mémoire

La TVI dans le code

```
SECTION INTVEC
```

```
B main  
B undefInterrupt ; Instruction indéfinie  
B softInterrupt ; Interruption logicielle
```

Ici, nous supportons **3 interruptions**

```
SECTION CODE
```

```
undefInterrupt  
; routine de traitement de l'interruption  
"instruction indéfinie"
```

Routine pour instruction indéfinie

```
softInterrupt  
; routine de traitement de l'interruption  
"interruption logicielle"
```

Routine pour interruption logicielle

```
main  
; routine de traitement de l'interruption  
"reset", donc, notre code principal
```

Code principal

```
SECTION DATA
```

Variables en mémoire

Table des vecteurs d'interruption

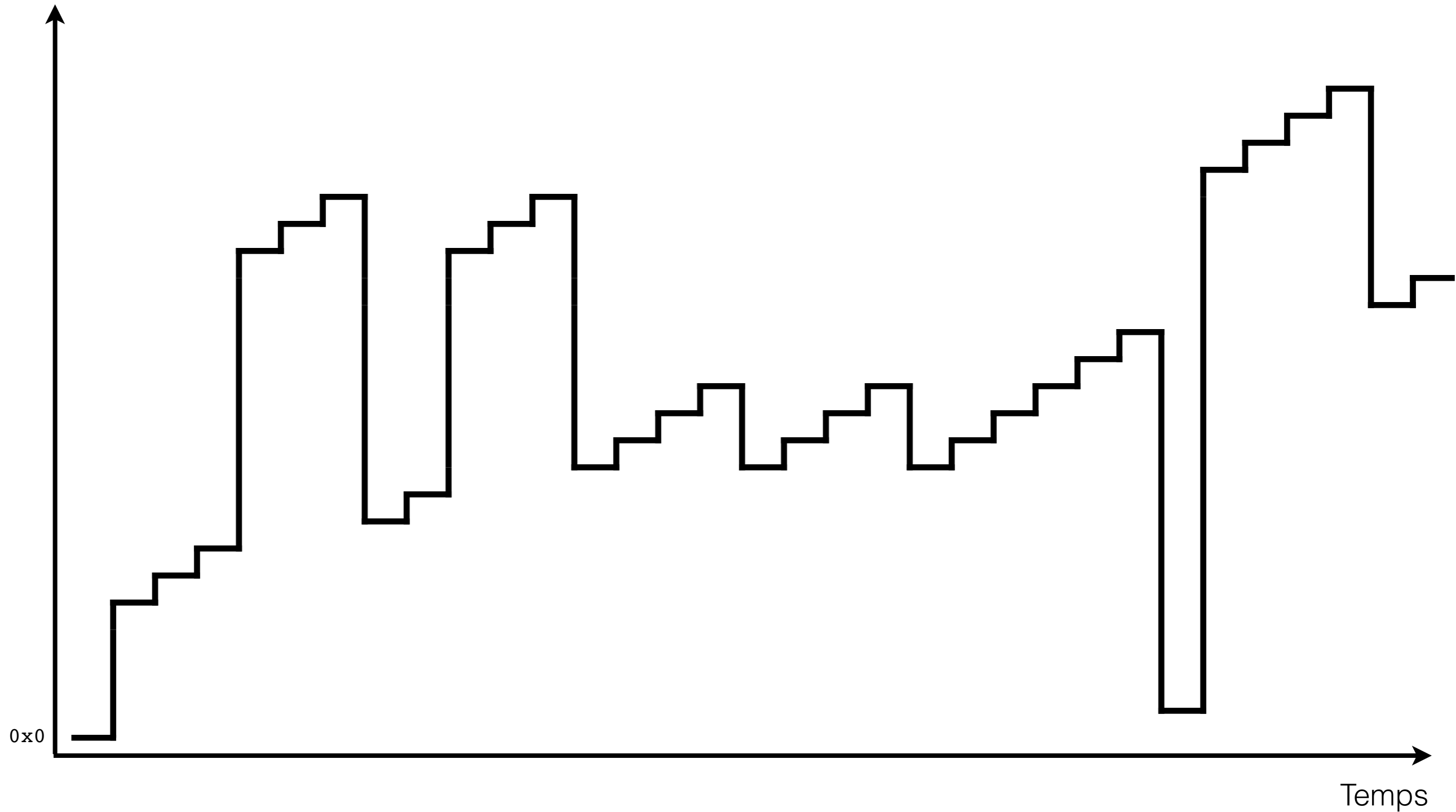
- Contient une instruction (en ARM) qui branche vers la routine de traitement de l'interruption
- Commence à l'adresse 0x0 de la mémoire
 - peut être déplacée ailleurs
 - dans un système complet, la table est modifiée par le système d'exploitation.

Démonstration

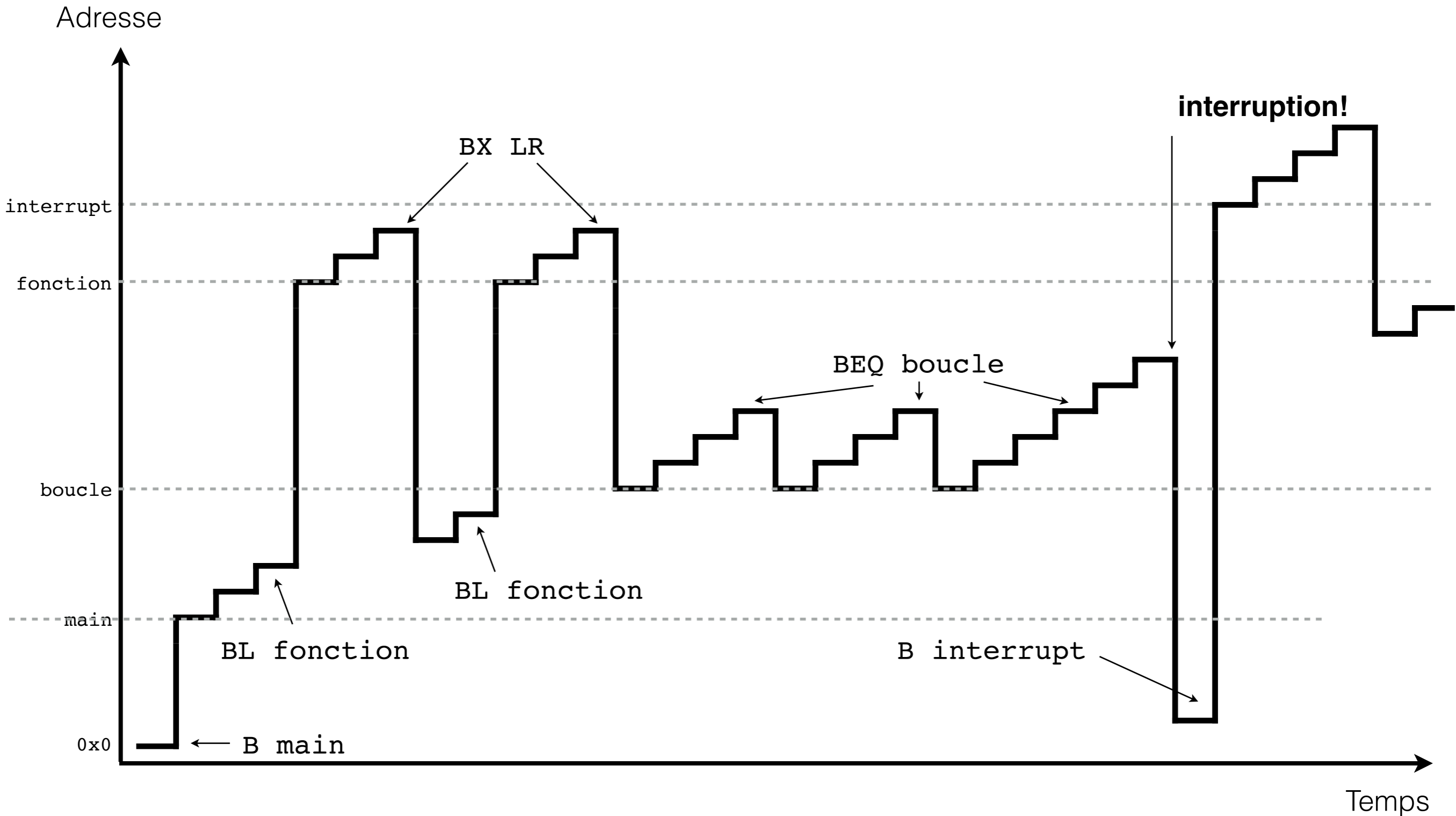
(Table des vecteurs d'interruption)

Évolution de PC...

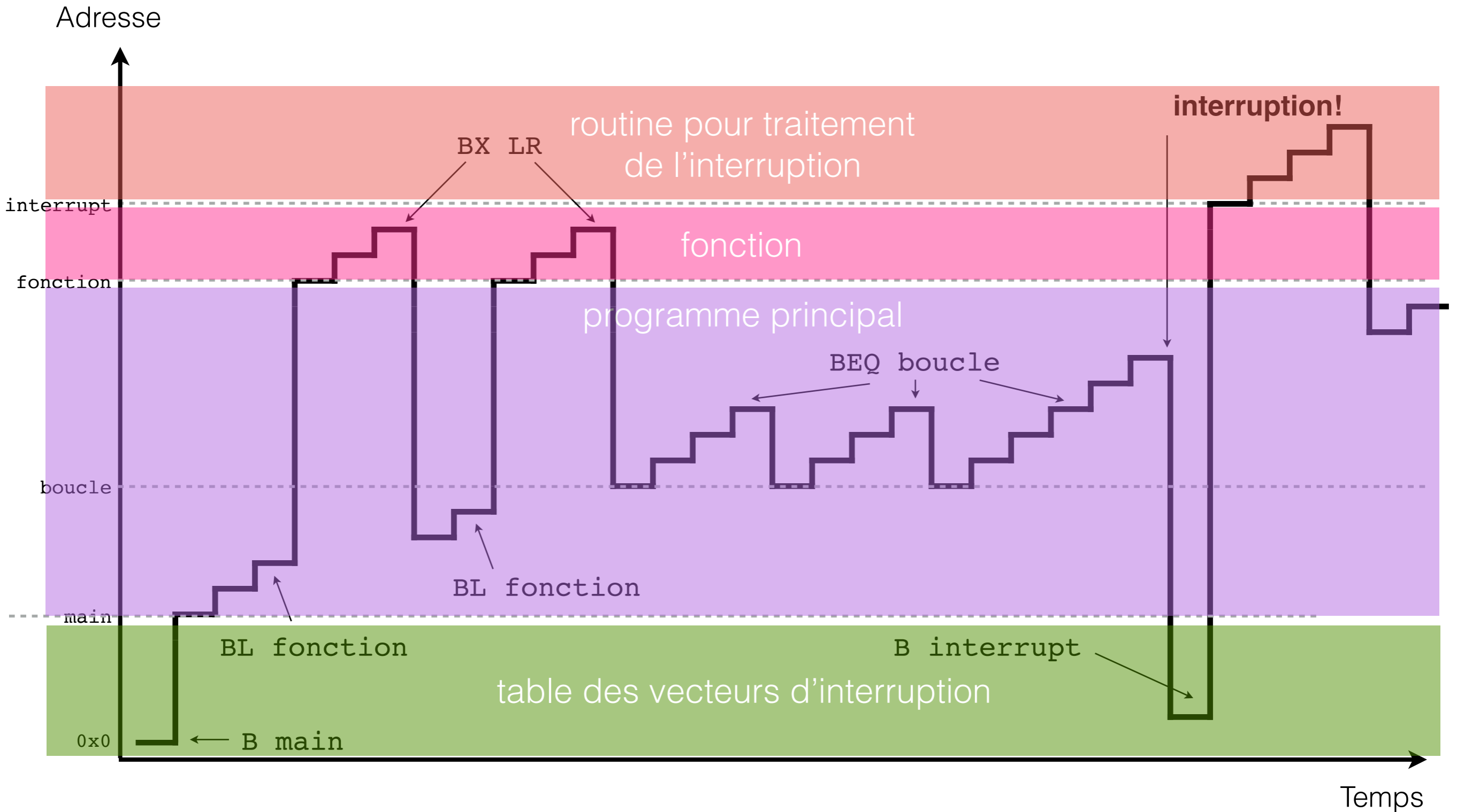
Adresse



Évolution de PC...



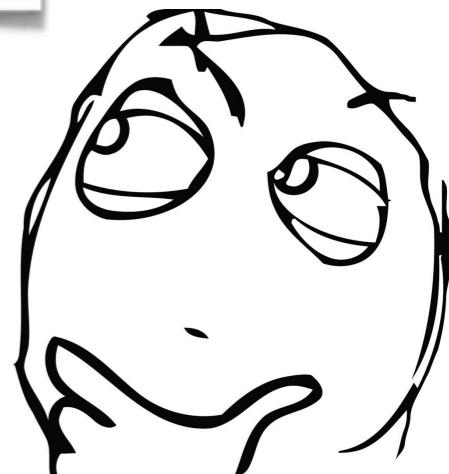
Évolution de PC...



Rappel: registres

- 16 registres de 32 bits:
 - R0 à R12: registres généraux
 - R13 à R15: registres spécifiques (ayant des fonctionnalités pré-établies)
 - R13: Pointeur de pile (*Stack Pointer* ou SP)
 - R14: Registre de liens (*Link Register* ou LR)
 - R15: Compteur de programme (*Program Counter* ou PC)
- 1 registre d'état, le *Current Program Status Register* (CPSR)
 - mémorise les résultats d'opérations arithmétiques
 - stocke le « mode d'exécution » (~~on y reviendra bientôt!~~)

On y revient maintenant!



Le «mode» du microprocesseur

- Un microprocesseur possède plusieurs modes
- En temps normal, le microprocesseur est en mode utilisateur.
- Lorsqu'une interruption survient, le microprocesseur adopte le mode correspondant à l'interruption traitée

Les modes en ARM

- Le microprocesseur ARM possède 6 modes, qui correspondent aux interruptions disponibles:
 1. **User**: le mode « par défaut » disponible pour le programmeur
 2. **Supervisor**: activé par une instruction du programmeur pour utiliser les fonctions systèmes (par exemple pour communiquer avec les périphériques);
 3. **Abort**: activé lors d'un accès mémoire invalide;
 4. **Undefined**: activé lors d'une erreur dans le décodage d'une instruction (instruction invalide)
 5. **IRQ**: activé par un périphérique (**I**nterrupt **ReQ**uest)
 6. **FIQ**: activé par un périphérique, mais qui peut être traité plus rapidement qu'une IRQ (**F**ast **I**nterrupt **reQ**uest)

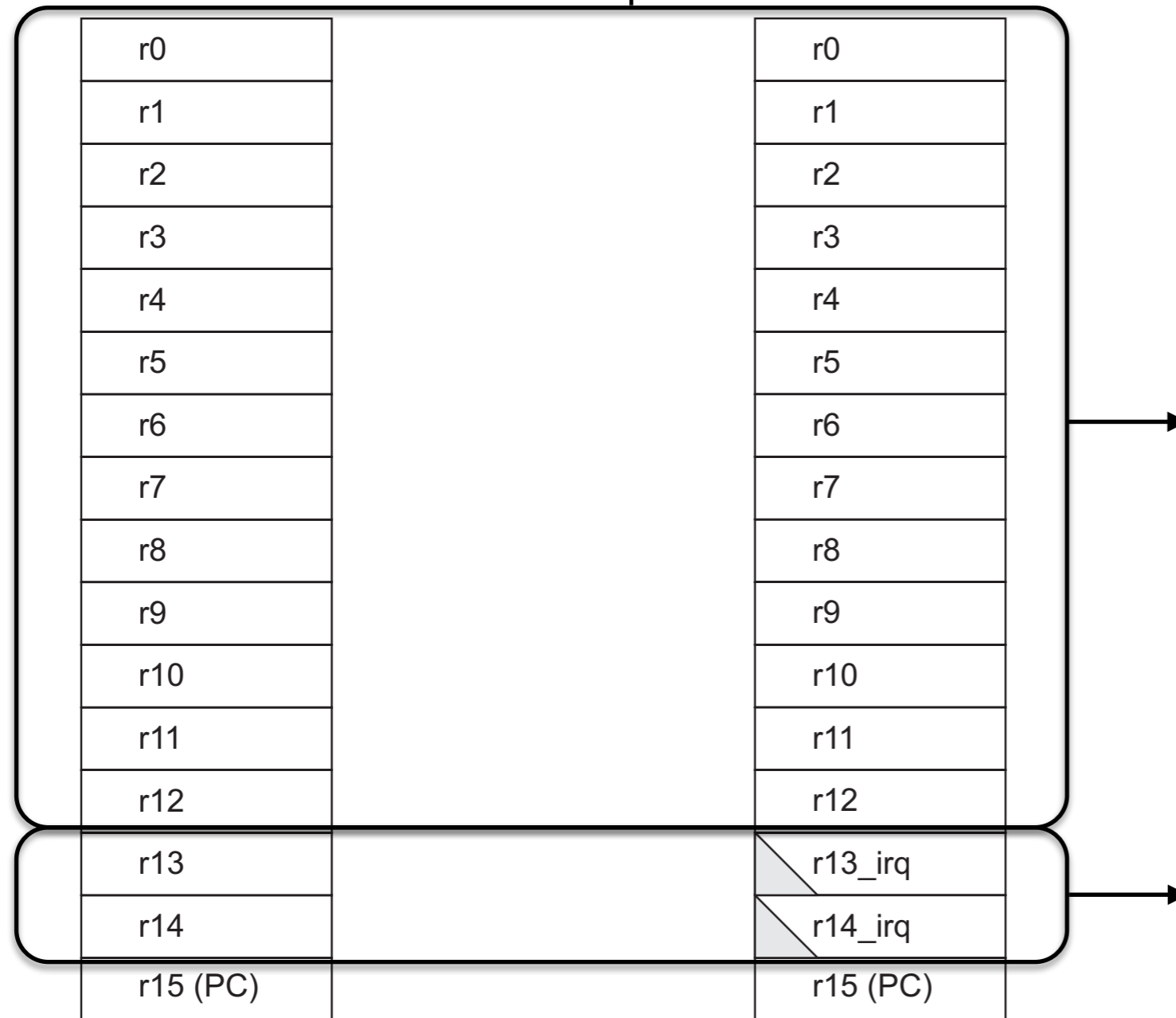
Le «mode» du microprocesseur

- Chaque mode possède ses registres qui lui sont propres
 - le microprocesseur ARM vu dans le cours possède 31 registres physiques en tout!
 - seulement 16 sont accessibles à la fois

Registres ARM

Registres généraux

Registres
pour le mode IRQ



Ces registres
sont partagés
entre les deux modes

Ces registres
ne sont pas partagés
entre les deux modes

Il y a donc au moins **2 registres physiques** de plus (R13_irq et R14_irq)!

Registres ARM

Registres généraux

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13
r14
r15 (PC)

Registres
pour le mode IRQ

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13_irq
r14_irq
r15 (PC)

Registres
pour le mode FIQ

r0
r1
r2
r3
r4
r5
r6
r7
r8_fiq
r9_fiq
r10_fiq
r11_fiq
r12_fiq
r13_fiq
r14_fiq
r15 (PC)

Il y a donc au moins **9 registres physiques** de plus
(R13_irq, R14_irq, R8_fiq, ..., R14_fiq)!

Registres ARM

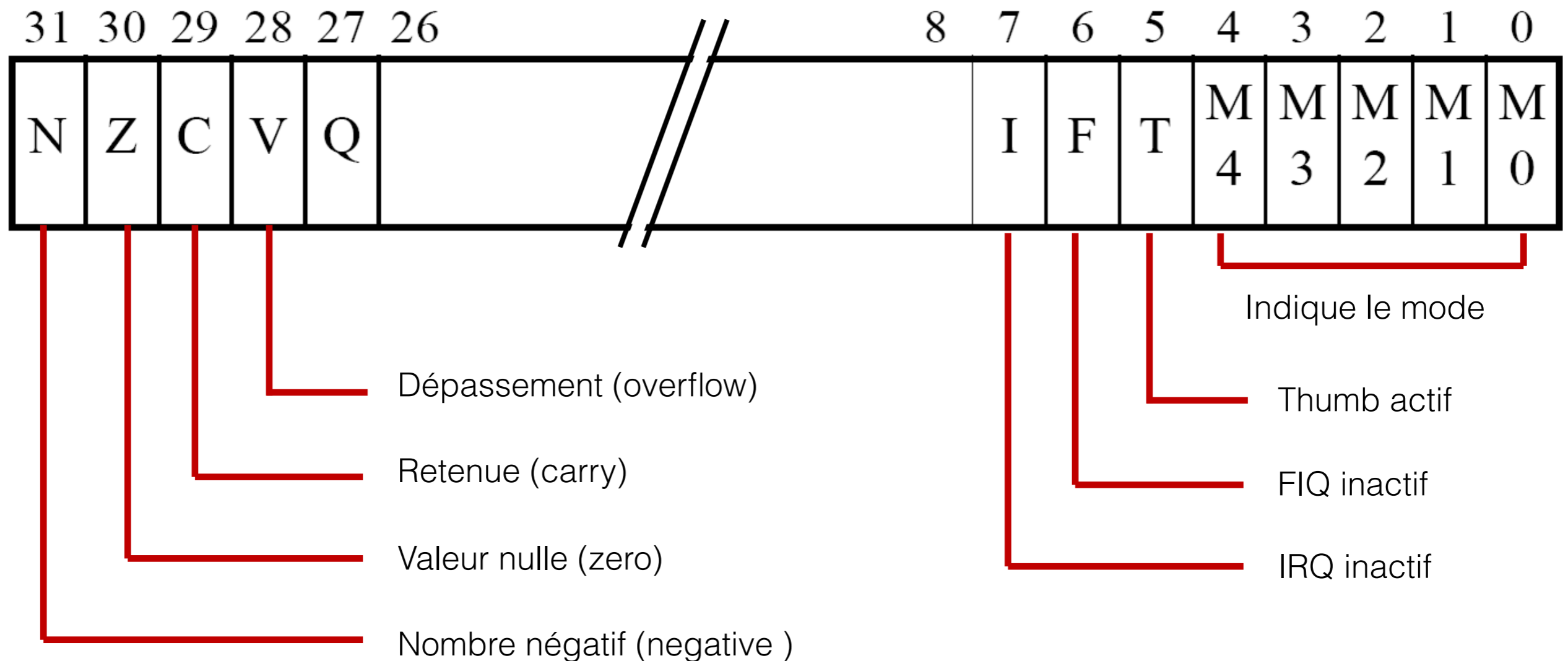
Pourquoi **Reset** n'est pas là?

System and User	FIQ	Supervisor	Abort	IRQ	Undefined
r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r2	r2
r3	r3	r3	r3	r3	r3
r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7
r8	r8_fiq	r8	r8	r8	r8
r9	r9_fiq	r9	r9	r9	r9
r10	r10_fiq	r10	r10	r10	r10
r11	r11_fiq	r11	r11	r11	r11
r12	r12_fiq	r12	r12	r12	r12
r13	r13_fiq	r13_svc	r13_abt	r13_irq	r13_und
r14	r14_fiq	r14_svc	r14_abt	r14_irq	r14_und
r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)

Il y a donc **15 registres physiques** de plus (pour un total de 31)

Rappel: registre d'état (CPSR)

- Un registre d'état (dans l'ALU) qui décrit l'état du processeur



Registre de statut (CPSR)

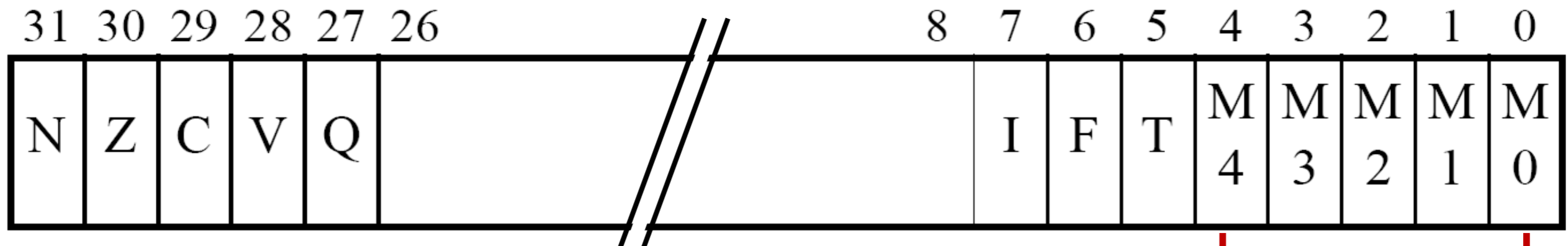
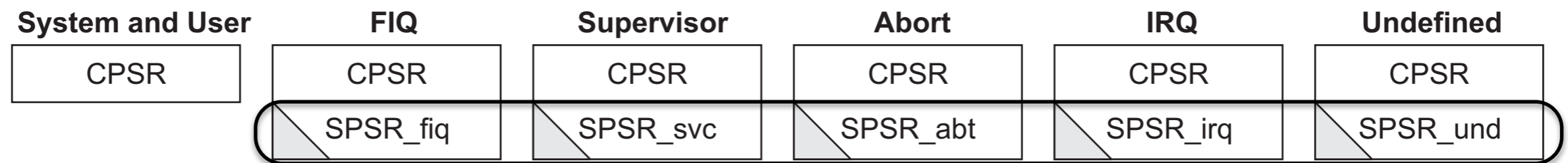


Table 2-2 PSR mode bit values

M[4:0]	Mode	Visible Thumb-state registers	Visible ARM-state registers
10000	User	r0-r7, SP, LR, PC, CPSR	r0-r14, PC, CPSR
10001	FIQ	r0-r7, SP_fiq, LR_fiq, PC, CPSR, SPSR_fiq	r0-r7, r8_fiq-r14_fiq, PC, CPSR, SPSR_fiq
10010	IRQ	r0-r7, SP_irq, LR_irq, PC, CPSR, SPSR_irq	r0-r12, r13_irq, r14_irq, PC, CPSR, SPSR_irq
10011	Supervisor	r0-r7, SP_svc, LR_svc, PC, CPSR, SPSR_svc	r0-r12, r13_svc, r14_svc, PC, CPSR, SPSR_svc
10111	Abort	r0-r7, SP_abt, LR_abt, PC, CPSR, SPSR_abt	r0-r12, r13_abt, r14_abt, PC, CPSR, SPSR_abt
11011	Undefined	r0-r7, SP_und, LR_und, PC, CPSR, SPSR_und	r0-r12, r13_und, r14_und, PC, CPSR, SPSR_und
11111	System	r0-r7, SP, LR, PC, CPSR	r0-r14, PC, CPSR

Registres d'état ARM



Il y a donc **5 registres d'état physiques** de plus
(pour un total de 6)

Démonstration

(registres bancaés)

Pile d'interruptions

- Une pile spéciale est disponible pour chaque type d'interruption
 - Il faut l'avoir préparée au préalable!
- On peut donc sauvegarder les registres avec PUSH et POP à l'intérieur des routines de traitement de l'interruption

System and User	FIQ	Supervisor	Abort	IRQ	Undefined
r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1
⋮	⋮	⋮	⋮	⋮	⋮
r11	r11_fiq	r11	r11	r11	r11
r12	r12_fiq	r12	r12	r12	r12
r13	r13_fiq	r13_svc	r13_abt	r13_irq	r13_und
r14	r14_fiq	r14_svc	r14_abt	r14_irq	r14_und
r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)

Traitement des interruptions

Interruption!

1. Terminer l'instruction en cours
2. Déterminer s'il faut traiter l'interruption.
- 3. Sauvegarder le contexte**
4. Déterminer l'adresse de la routine de traitement de l'interruption
5. Exécuter cette routine

Traitement de l'interruption...

1. Restaurer le contexte
2. Reprendre là où le processeur était rendu

Le contexte

- Le « contexte » d'un programme représente les informations importantes à son bon déroulement.
- Quelles sont ces informations?
 - PC (notre vieil ami)
 - Les drapeaux dans le CPSR
 - Les registres

Le contexte

- Qu'arrive-t-il au contexte lors d'une interruption?
 - PC (notre vieil ami)
 - est sauvegardé automatiquement dans LR ←
 - Le CPSR
 - est sauvegardé automatiquement dans le SPSR ←
 - Les registres
 - dépend du mode!

Quel LR?

Le LR correspondant au mode de l'interruption soulevée!

Quel SPSR?

Le SPSR correspondant au mode de l'interruption soulevée!

Traitement des interruptions

Interruption!

1. Terminer l'instruction en cours
2. Déterminer s'il faut traiter l'interruption.
3. Sauvegarder le contexte
4. Déterminer l'adresse de la routine de traitement de l'interruption
- 5. Exécuter cette routine**

Traitement de l'interruption...

1. Restaurer le contexte
2. Reprendre là où le processeur était rendu

Utilisation d'un registre

- Dans la routine de traitement de l'interruption, on peut simplement utiliser les registres comme d'habitude

- Par exemple:

```
fiqInterrupt
; routine de traitement de l'interruption fiq

MOV R8, #1    ; on utilise R8_fiq

MOV R0, #2    ; on utilise R0!

...
```

- Quand R8 est utilisé dans une routine de traitement d'une interruption **FIQ**, c'est en fait R8_fiq qui est utilisé
- Cependant, quand R0 est utilisé, c'est R0! Quel est le problème?
 - Si notre programme principal avait besoin de R0, on vient de changer sa valeur!

Traitement des interruptions

Interruption!

1. Terminer l'instruction en cours
2. Déterminer s'il faut traiter l'interruption.
3. Sauvegarder le contexte
4. Déterminer l'adresse de la routine de traitement de l'interruption
5. Exécuter cette routine

Traitement de l'interruption...

- 1. Restaurer le contexte**
- 2. Reprendre là où le processeur était rendu**

Restauration du contexte

- Qu'arrive-t-il au contexte lors d'une interruption?
 - PC (notre vieil ami)
 - était sauvegardé automatiquement dans LR
 - doit être restauré à partir de la valeur dans LR
 - Les drapeaux du CPSR
 - étaient sauvegardés automatiquement dans **SPSR**
 - sont restaurés automatiquement dans CPSR
 - Les registres
 - dépend du mode!

Reprise du programme

- Dans plusieurs architectures, une instruction spéciale est utilisée pour indiquer la fin d'une interruption
- En ARM, c'est, comment dire, un peu bizarre... il faut utiliser une instruction
 - qui change les drapeaux (avec **S**)
 - et qui stocke son résultat dans **PC** (!)

```
fiqInterrupt
; routine de traitement de l'interruption FIQ

...

; terminé!
SUBS PC, LR, #4
```

Dans une FIQ, on sauvegarde PC directement (donc l'adresse de l'instruction courante + 8).

Donc, pour revenir à l'instruction suivante, il faut placer LR - 4 dans PC.

Démonstration

(sauvegarde du contexte)

Résumé

Interruption!

1. Terminer l'instruction en cours
2. Déterminer s'il faut traiter l'interruption.
3. Sauvegarder le contexte
4. Déterminer l'adresse de la routine de traitement de l'interruption
5. Exécuter cette routine

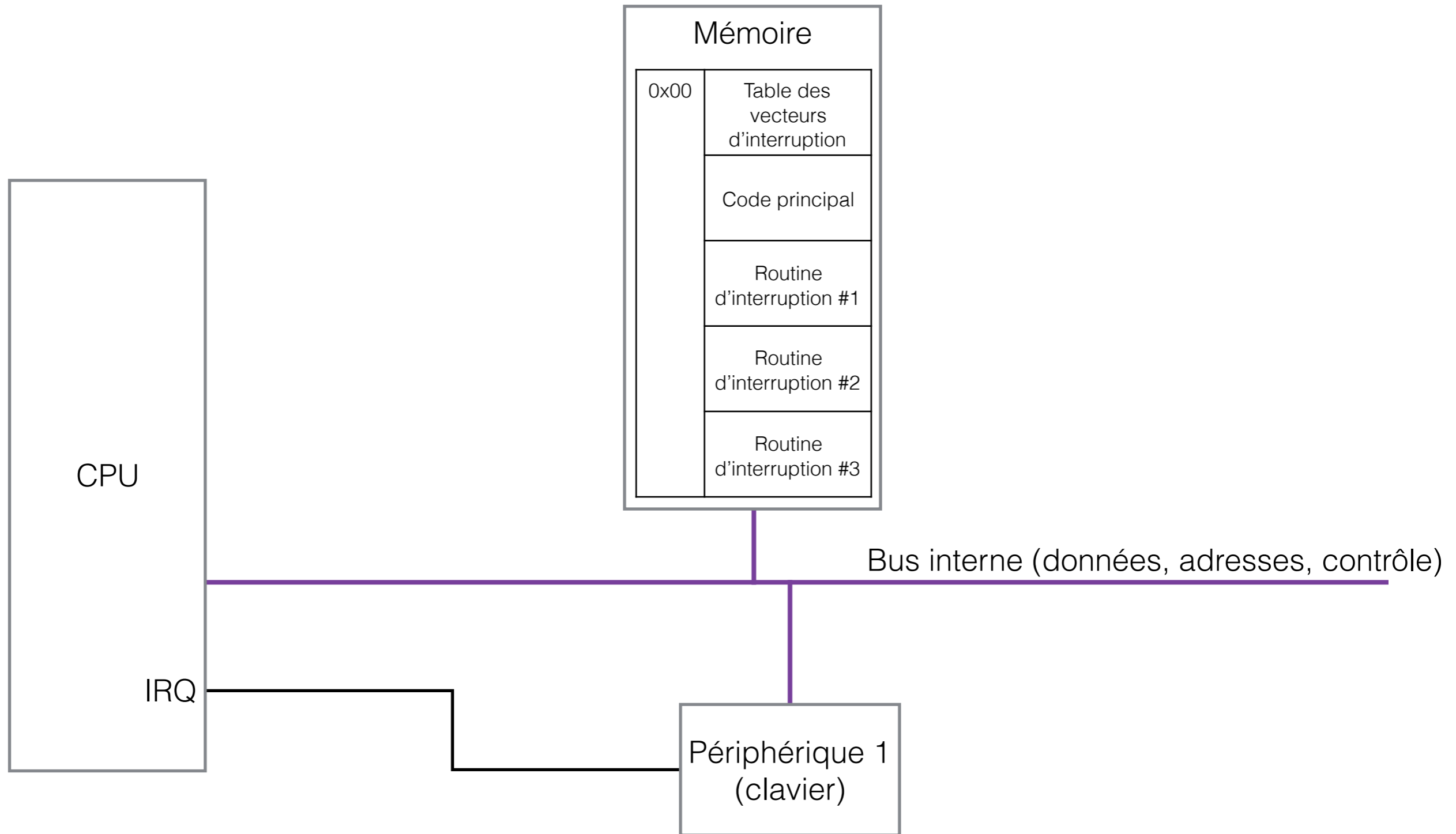
Traitement de l'interruption...

1. Restaurer le contexte
2. Reprendre là où le processeur était rendu

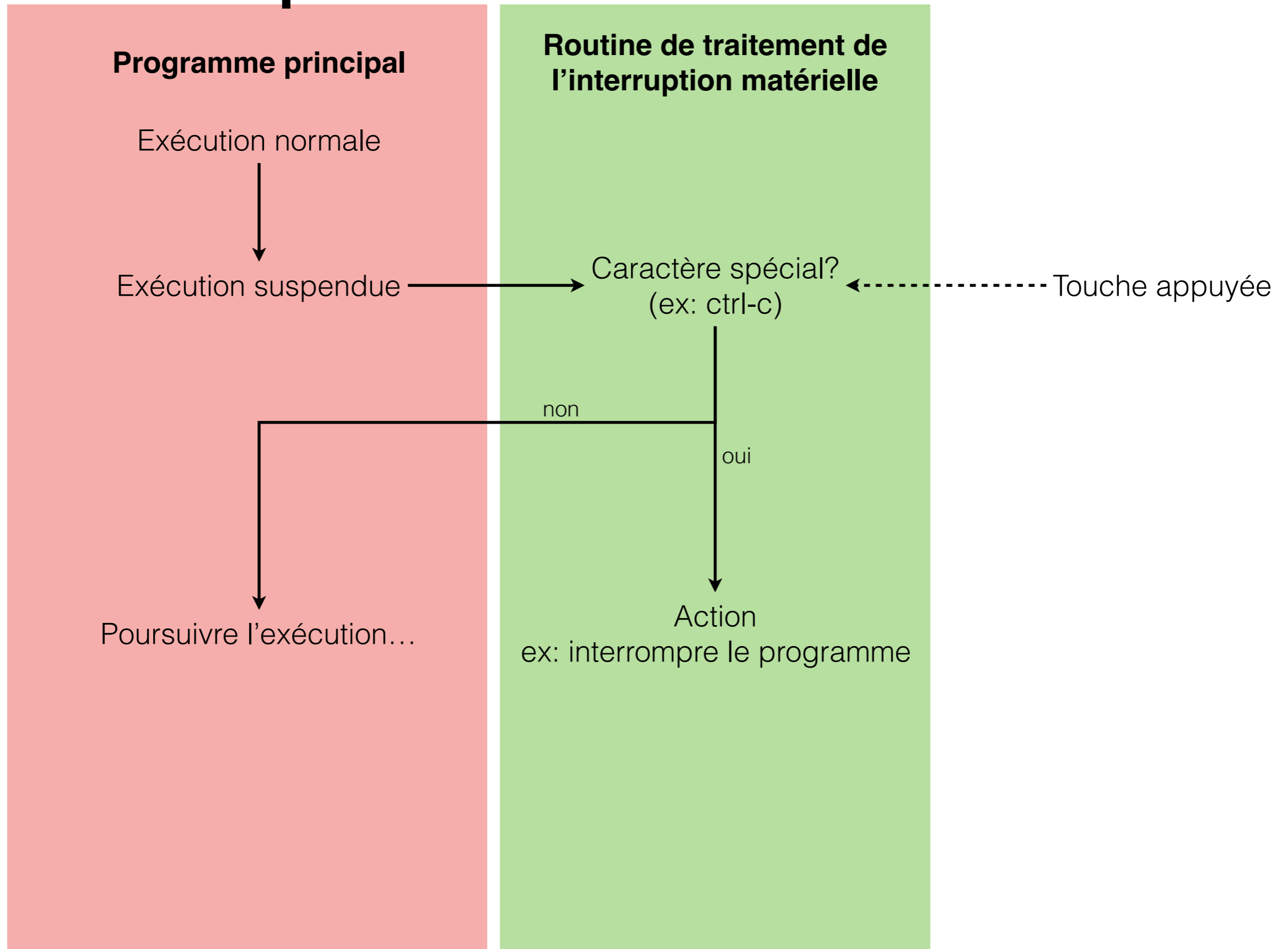
Types d'interruptions

- **Systeme**: reset, faute matérielle générale, etc.
- **Exception**: erreur au cours de l'exécution des instructions, par exemple:
 - instruction invalide
 - division par 0
 - tentative d'accès à de la mémoire protégée
- **Matérielles**: générées par les périphériques, par exemple:
 - le clavier dans notre scénario 1 plus haut
 - l'imprimante lorsqu'elle a terminé, etc.
- **Logicielles**: générées par le programmeur avec une instruction spéciale (présente dans tous les jeux d'instructions)

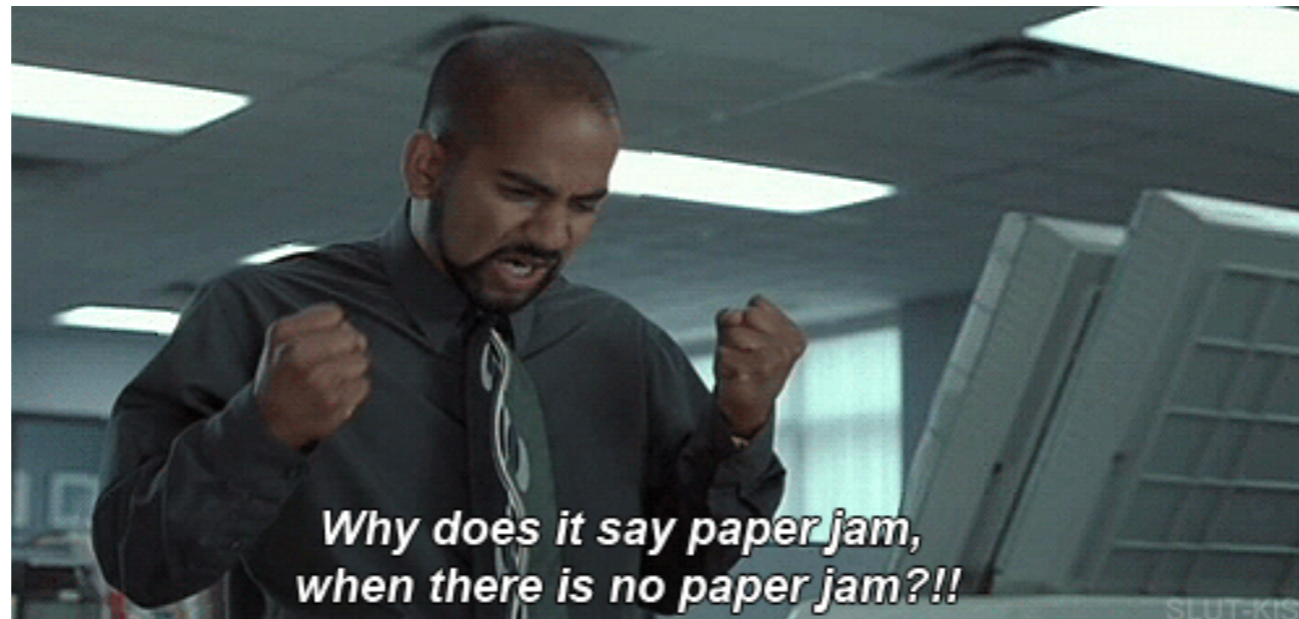
Interruptions matérielles (périphériques)



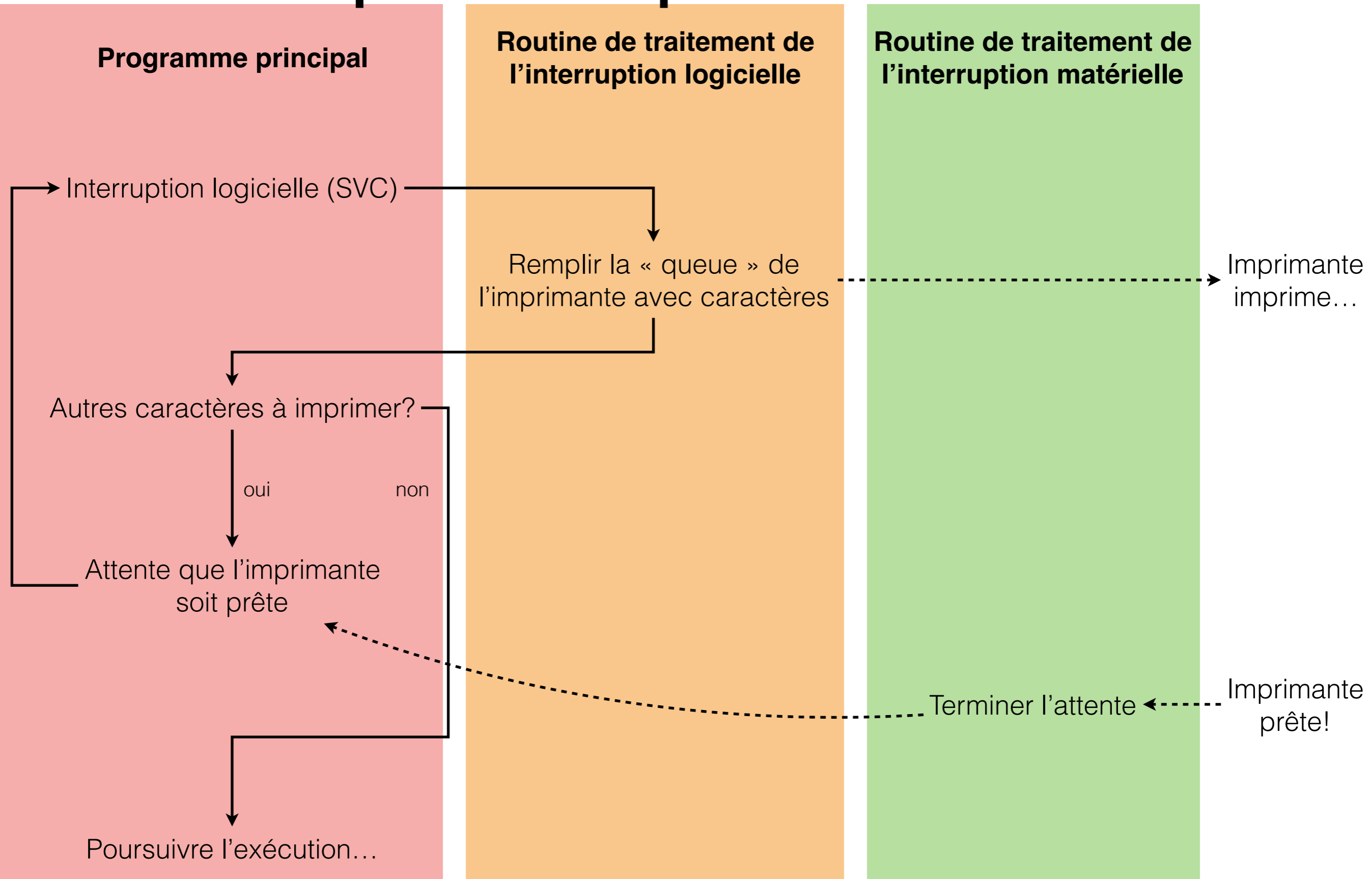
Interruption clavier



Interruption imprimante



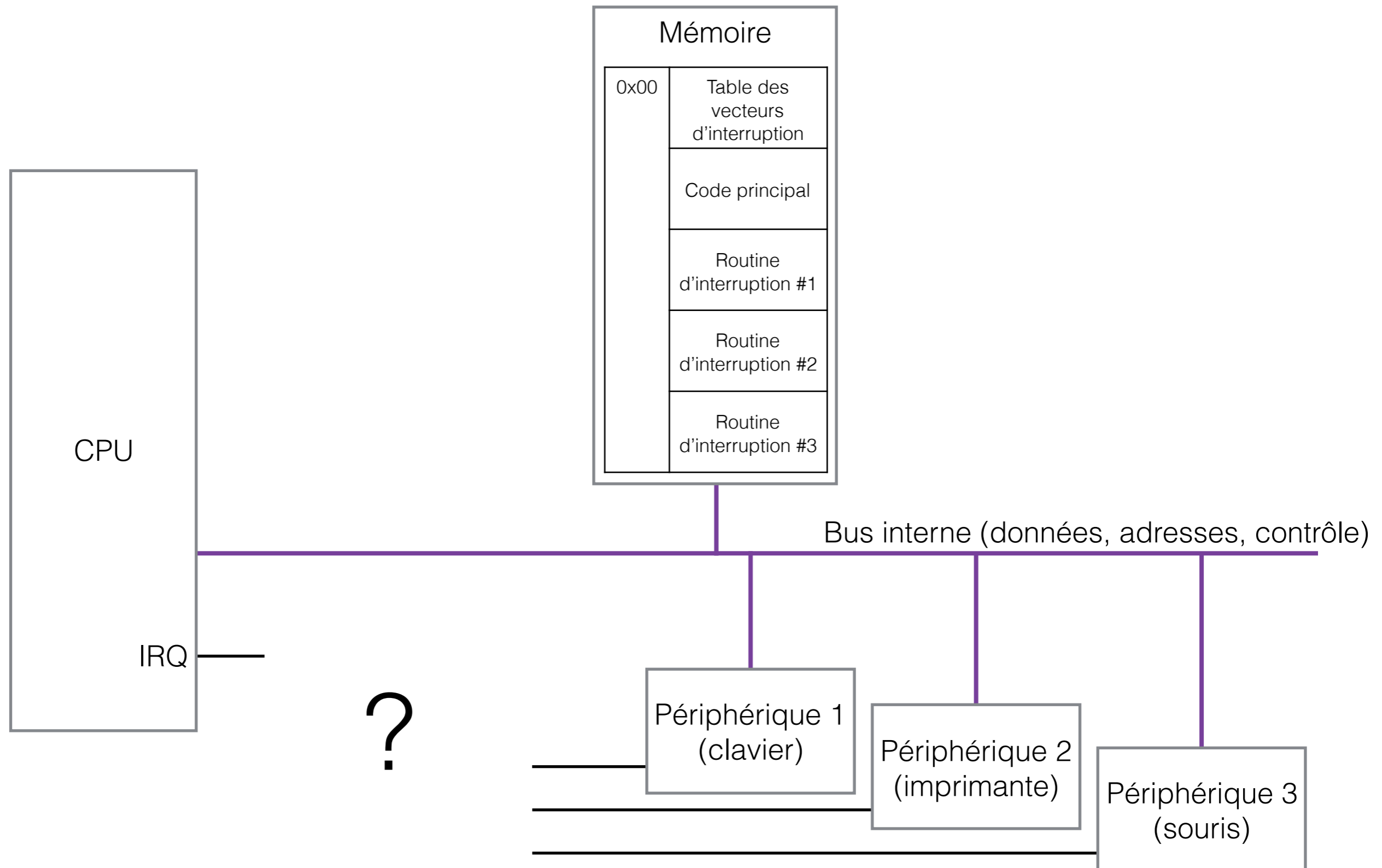
Interruption imprimante



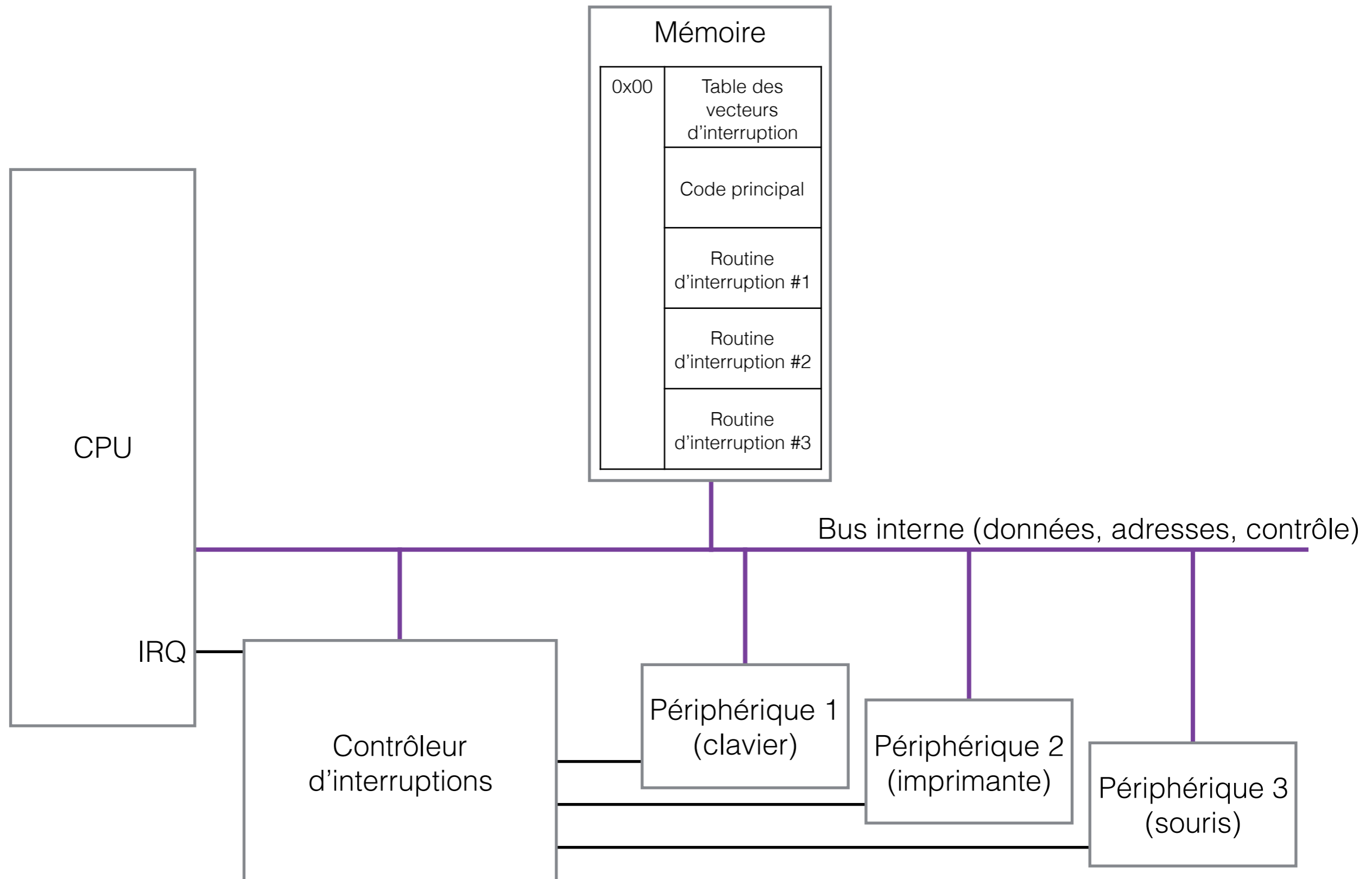
Démonstration

(Simulation d'une interaction avec imprimante)

Interruptions matérielles (périphériques)



Contrôleur d'interruptions



Contrôleur d'interruptions (NVIC)

- Les interruptions de l'ordinateur sont gérées par le contrôleur d'interruption.
- Le contrôleur d'interruptions:
 - reçoit les signaux d'interruptions
 - peut activer (masquer) ou désactiver certaines interruptions.
 - modifier la priorité des interruptions.
 - signale les interruptions au microprocesseur à l'aide de fils dédiés à cette fin.
 - peut être configuré via des instructions dans la mémoire
- Dans le cas du processeur ARM, le contrôleur d'interruptions est inclus dans le cœur.

Types d'interruptions

- **Systeme**: reset, faute matérielle générale, etc.
- **Exception**: le processeur peut générer des interruptions s'il n'est pas capable de lire ou d'exécuter une instruction (opcode invalide, division par 0, mémoire protégée, etc).
- **Matérielles**: générées par les périphériques
- **Logicielles**: il y a une instruction qui permet de générer une interruption dans tous les jeux d'instructions.
 - En ARM, c'est l'instruction SVC (Service Call)

Interruptions du système

- L'interruption **reset** est l'interruption système la plus prioritaire. Cette interruption peut survenir pour plusieurs raisons:
 - mise sous tension, activation de la broche reset du microprocesseur, instruction reset, etc.
- Lors d'un reset, toutes les autres interruptions sont ignorées

Exceptions

- Les **exceptions** surviennent quand un évènement logiciel spécial arrive. Par exemple:
 - instruction invalide
 - division par 0
 - référence à une adresse invalide
 - accès invalide à une adresse protégée
- Les exceptions ont un très haut niveau de priorité parce le microprocesseur est dans une impasse: il ne peut exécuter l'instruction en cours en raison d'une erreur de programmation!

Interruptions matérielles

- Les interruptions **matérielles** sont générées par les périphériques
- La plupart des périphériques ont une ligne de contrôle reliée au contrôleur d'interruptions qui leur permet de signaler un événement.
- Lors d'une interruption de périphérique, le microprocesseur obtient automatiquement le # de l'interruption du contrôleur et utilise ce numéro pour trouver l'ISR à exécuter à partir de la table des vecteurs d'interruptions.

Interruptions logicielles

- Les interruptions **logicielles** sont des interruptions « provoquées » par le programmeur. Le programmeur utilise une instruction qui déclenche une interruption.
- Les interruptions logicielles ont un effet similaire à un appel de fonction avec une différence fondamentale:
 - l'adresse de la fonction appelée est dans la table des vecteurs d'interruption plutôt qu'être une adresse relative au programme.
- Les interruptions logicielles servent souvent à appeler des fonctions du système d'exploitation dont l'adresse est inconnue du programmeur, mais gérée par le système d'exploitation (grâce à la table des vecteurs d'interruption).

Interruptions et système d'exploitation

- 2 utilités principales:
 - accès aux périphériques
 - exécution de plusieurs processus

Interruptions et système d'exploitation

- Accès aux périphériques
 - Chaque périphérique (ex: clavier, imprimante) peut se comporter légèrement différemment des autres
- C'est le système d'exploitation (*Operating System*, ou *OS*) qui rend les programmes « indépendants » du matériel. Comment?
 - C'est l'OS qui modifie les routines de traitement des interruptions à exécuter en fonction du matériel branché dans l'ordinateur (via la table des vecteurs d'interruption)
- Les programmes peuvent donc utiliser la table des vecteurs d'interruption comme d'habitude

Accès aux périphériques

- Exemple: recevoir une valeur du clavier
 - interruption matérielle (c'est l'utilisateur qui tape au clavier)
- Exemple: envoyer des données à l'imprimante
 - interruption logicielle (c'est notre programme qui veut imprimer)
- Le système d'exploitation permet au même programme de « parler » à plusieurs modèles de claviers et d'imprimantes

Interruptions et système d'exploitation

- Les interruptions permettent l'exécution de plusieurs processus
- Comment?
 - Une horloge génère des interruptions périodiquement
 - À chaque interruption, on change le processus à exécuter