Révision finale

GIF-1001 Ordinateurs: Structure et Applications, Hiver 2017

Jean-François Lalonde

Logistique — examen final

- 40% de la note totale
- Mardi 25 avril de 14h30 à 17h30
- Local: PLT-1112
- Une feuille 8.5 x 11, recto-verso, écrite *à la main*
- Droit à la calculatrice (approuvée)

Stratégies *avant* l'examen

- Dans les jours précédant l'examen:
  - DORMIR
  - Étude:
    - exercices facultatifs, exercices faits en classe
    - examens des années précédentes
    - lecture: notes de cours, livre
    - objectifs du cours (sur le site web)
- Dans les heures précédant l'examen:
  - Alimentation adéquate (attention au «sugar crash»)
  - Économisez votre énergie intellectuelle
  - Détente

Stratégies *durant* l'examen

- *Avant* de commencer:
  - Compter les pages
  - Arracher la feuille des annexes pour un accès facile
  - Survoler l'examen et classer les questions selon leur difficulté/temps
- *Après* avoir commencé:
  - Commencer par les questions les plus faciles!
  - Bloqué? Sautez la question et prenez note d'y revenir
  - Ne pas oublier de respirer

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 ou le processeur était rendu

Le « contexte »

- Quelles sont les informations importantes au bon déroulement d'un programme?
  - PC (notre vieil ami)
    - est sauvegardé automatiquement dans LR
      - la valeur sauvegardée dans LR dépend du type d'interruption. Par exemple, dans une **FIQ**, on sauvegarde PC directement (donc l'adresse de l'instruction courante **+ 8**)
  - Les drapeaux de l'ALU
    - situés dans le « registre » spécial CPSR
    - est sauvegardé automatiquement dans un autre registre spécial: SPSR
  - Les registres
    - devront être sauvegardés à la mitaine, au besoin!

Interruptions et système d’exploitation

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

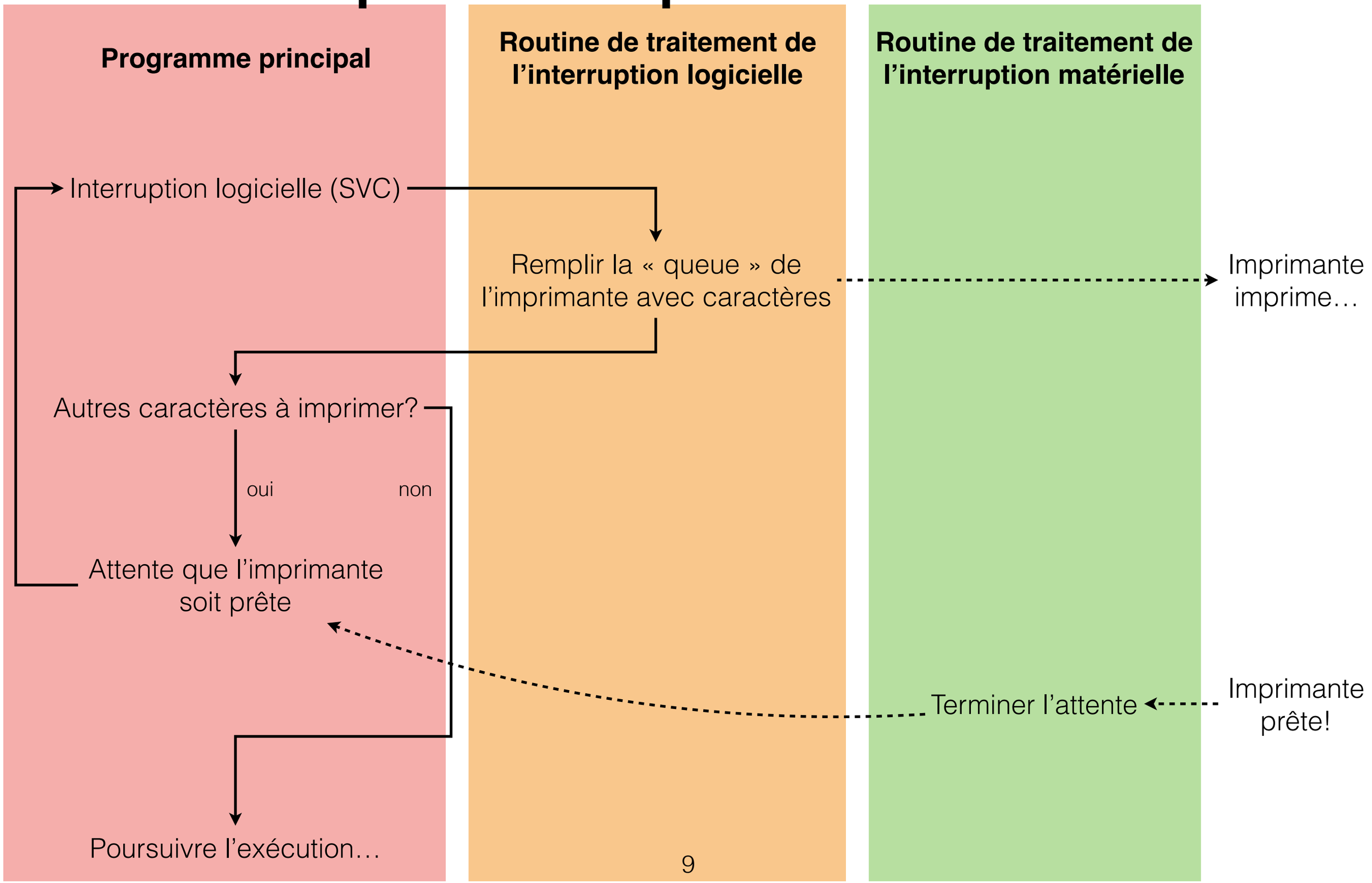
Registres ARM

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)

Interruption imprimante



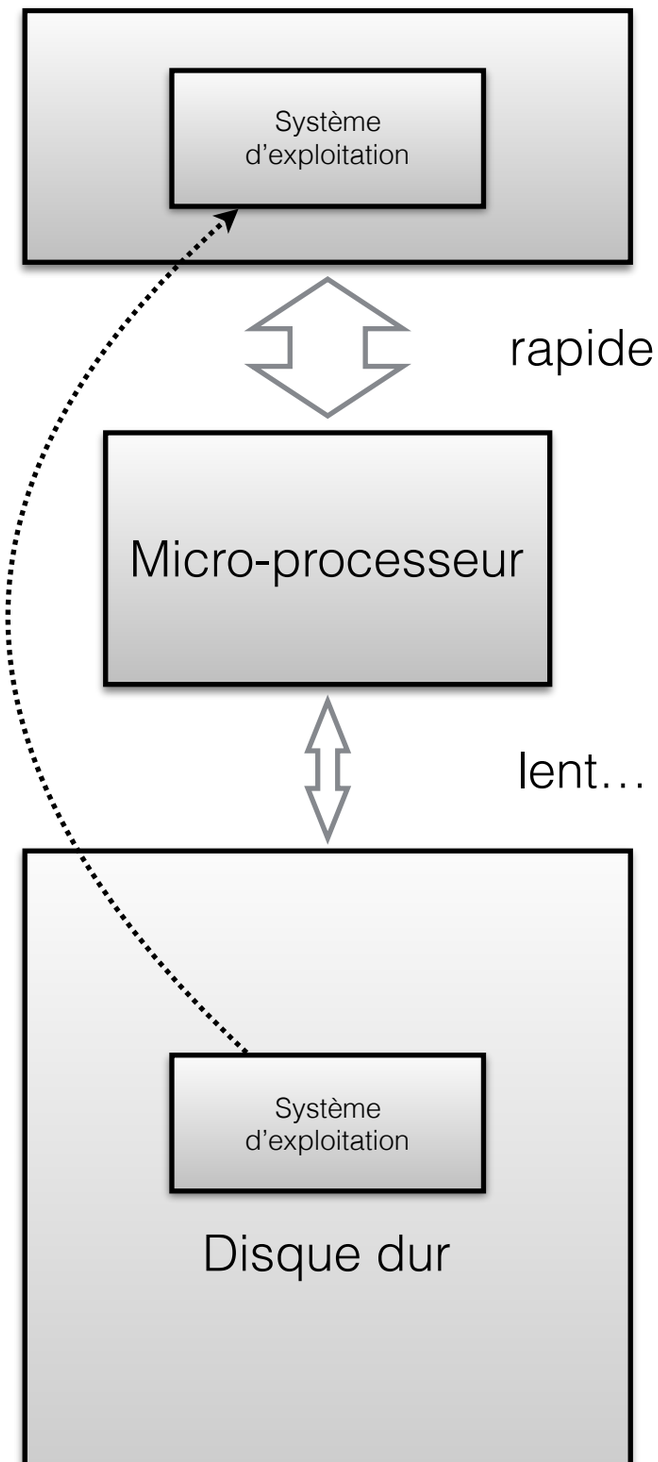
Démarrage: objectifs

- L'objectif de la séquence de démarrage est de démarrer le système d'exploitation (SE)

- Où est-il situé?
  - Sur le disque dur

- Peut-on l'exécuter s'il est sur le disque dur?
  - Non, le disque dur est un périphérique de stockage *lent*

- Donc, il nous faut le transférer dans la mémoire principale (RAM)



(SE = Système d'Exploitation)

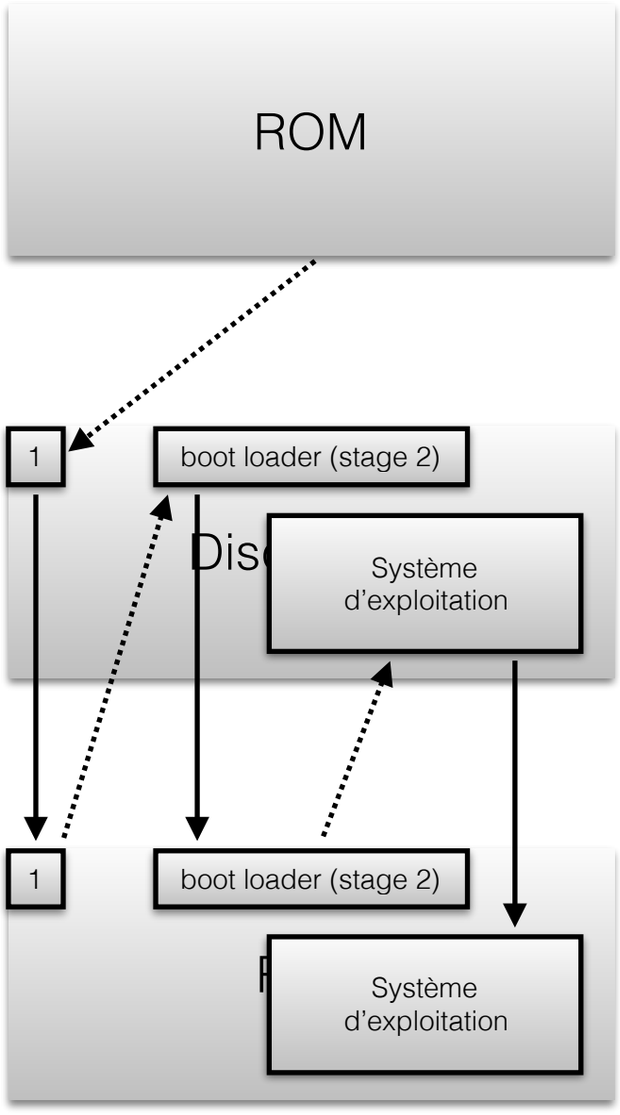
Démarrer un ordinateur—SE moderne

1. Exécution du BIOS (stocké dans la ROM)

2. Le BIOS charge le « boot loader stage 1 » situé dans le « Master Boot Record » (MBR) dans la RAM, et démarre l'exécution de ce programme

3. Le programme charge un autre programme, le « boot loader stage 2 » en mémoire, et démarre l'exécution de ce programme

4. Le « boot loader stage 2 » demande à l'utilisateur quoi faire (si désiré). Il charge le SE en mémoire, et démarre son exécution



Mémoire contigüe et paginée

- Mémoire:
  - contigüe
  - paginée
- Adresses physiques et virtuelles

Mémoire contigüe

1. Un nouveau programme est copié dans un emplacement disponible en mémoire, de façon contigüe.

2. On peut créer des partitions de taille:

    - **fixe**: la première partition disponible est choisie quand un nouveau processus doit être alloué
    - **variable**: on doit déterminer où créer la partition, nécessite le choix d'un algorithme d'allocation mémoire plus compliqué

3. Le programme utilise des adresses "virtuelles"

4. Le MMU traduit les adresse virtuelles en adresses physiques

Mémoire paginée, étape #1

- Diviser la mémoire **virtuelle** en « pages » de taille fixe
  - par exemple: 4Ko
- Diviser la mémoire **physique** en « trames » de *la même taille* que les pages
  - dans notre exemple: 4Ko!

Mémoire paginée, étape #2

- On divise une adresse en deux:
  - le numéro de page (MSB)
  - la position dans la page (LSB)
- Dans notre exemple, les pages ont 4Ko. Combien de bits avons-nous besoin pour encoder la position de chaque octet dans la page?
  - il y a 4K octets dans une page, donc $4K = 4 \times 2^{10}$
  - $\log_2(4 \times 2^{10}) = \log_2(2^2 \times 2^{10}) = \log_2(2^{12}) = 12$ bits

Quelle position dans quelle page réfère cette adresse?

0x 0 0 0 3 D 7 A 1

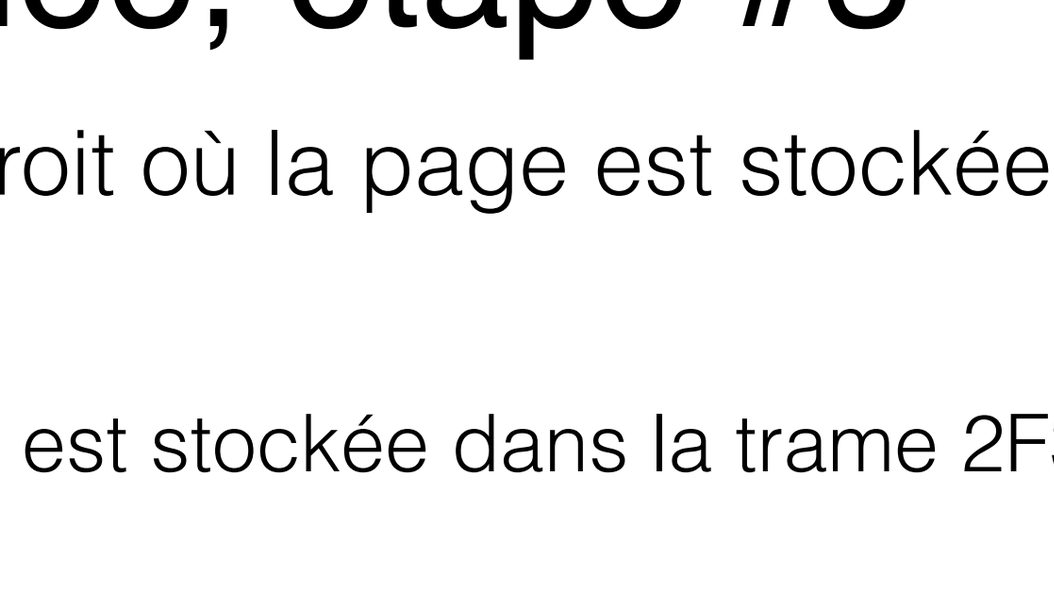
	Numéro de page					Position		
0x	0	0	0	3	D	7	A	1

Mémoire paginée, étape #3

- Si l'on vous donne l'endroit où la page est stockée en mémoire.
  - Par exemple: la page 3D est stockée dans la trame 2F3
- Comment faire pour convertir une adresse virtuelle en une adresse physique?

Adresse virtuelle

`0x 0 0 0 3 D 7 A 1`



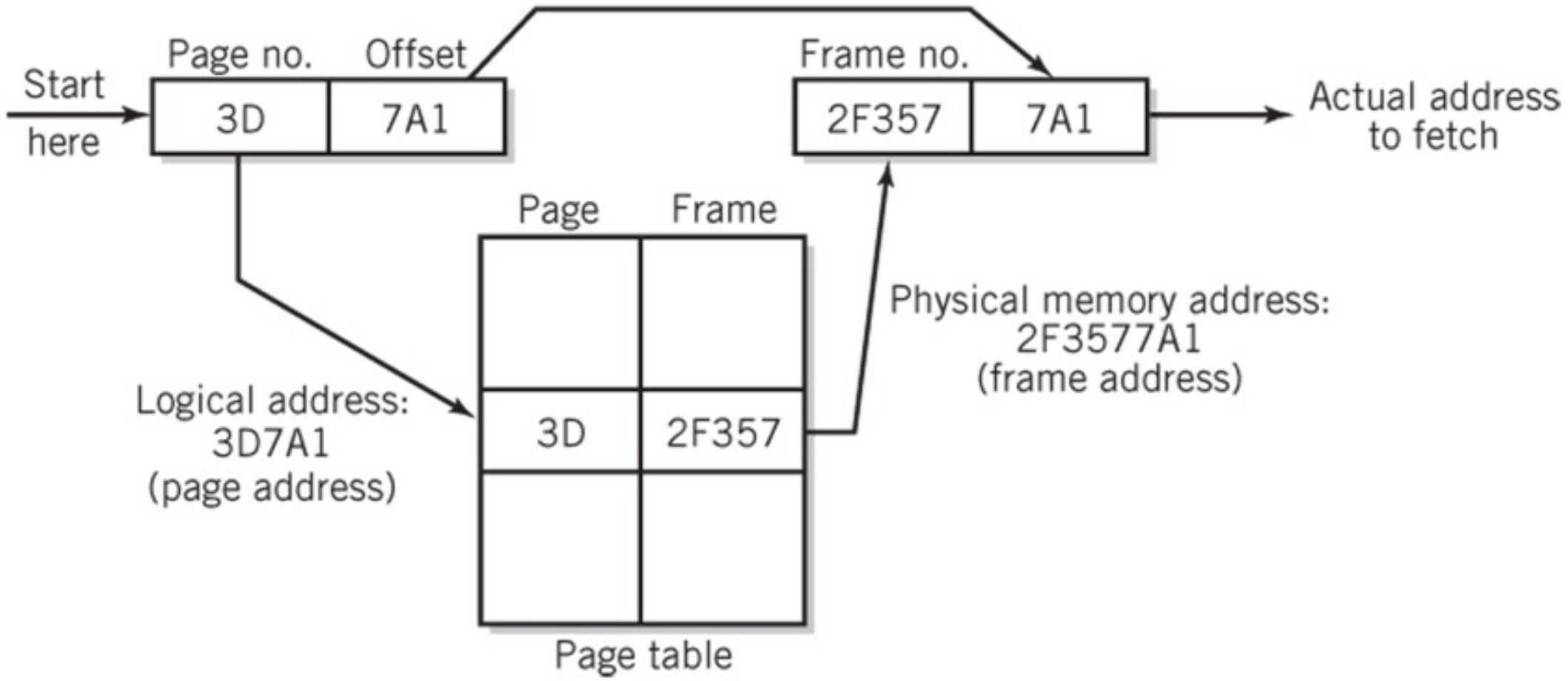
Adresse physique

`0x 0 0 2 F 3 7 A 1`

Mémoire paginée, étape #4

- Comment fait-on pour savoir
  - l'endroit où une page est stockée?
  - la correspondance entre une page et une trame?
- On utilise la table des pages!

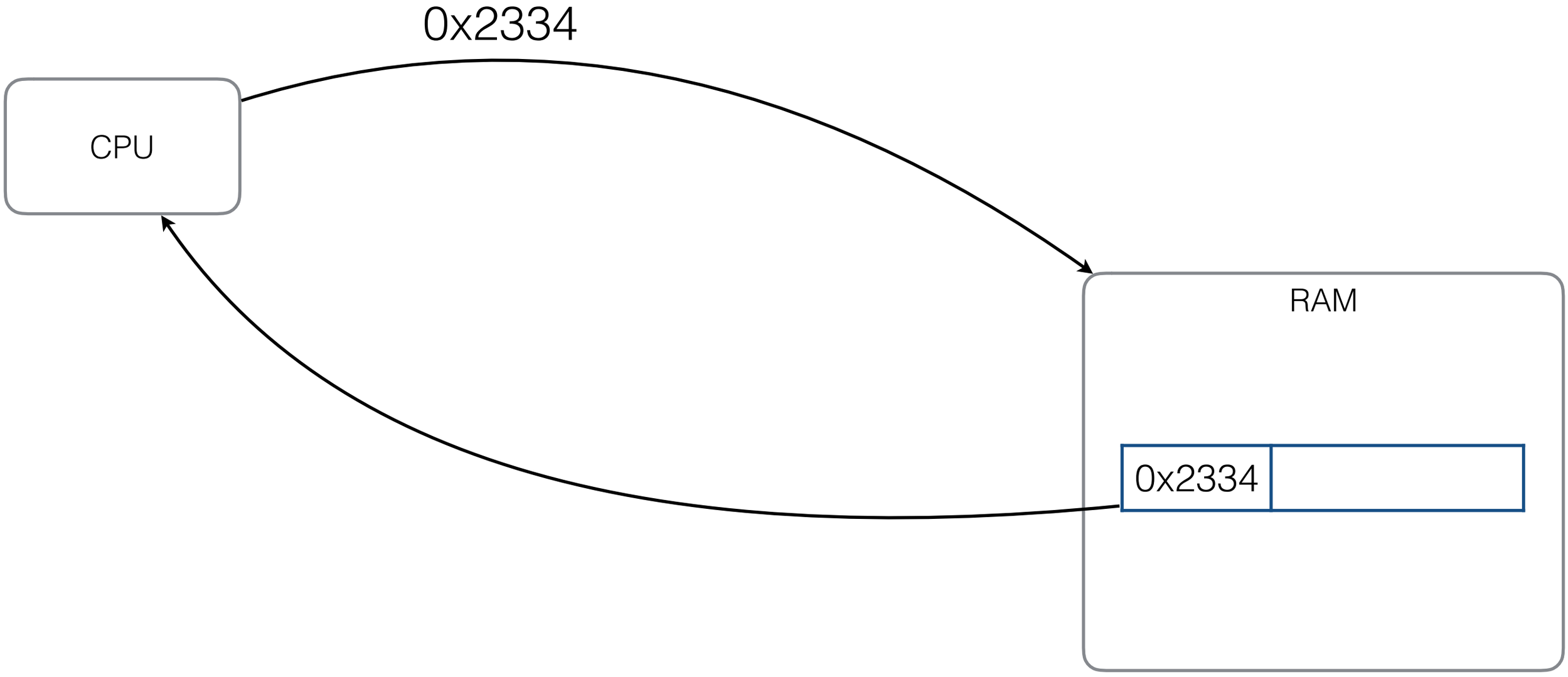
Table des pages



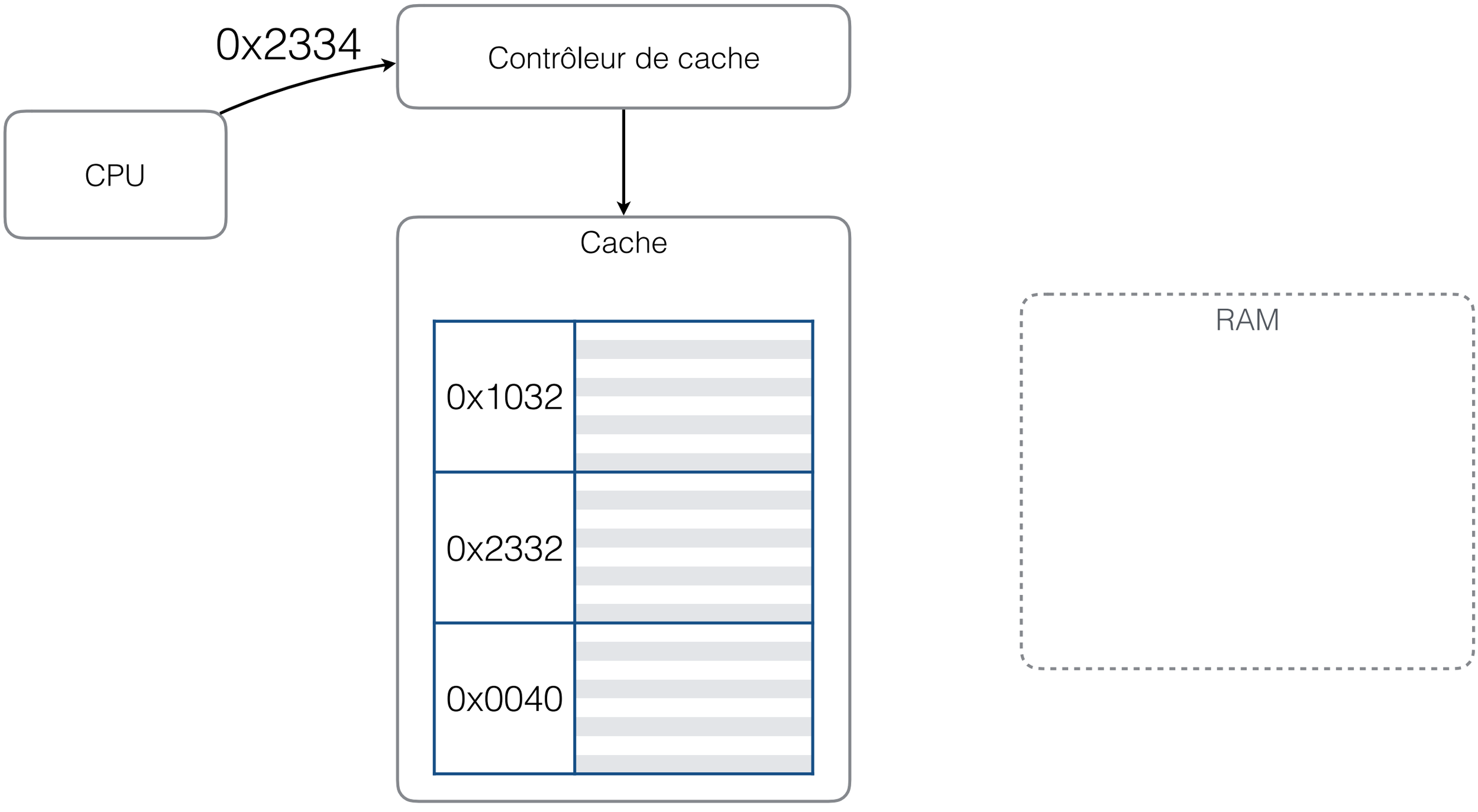
Englander, ch. 18

Lecture en mémoire

Comment faire pour accélérer les accès mémoire?

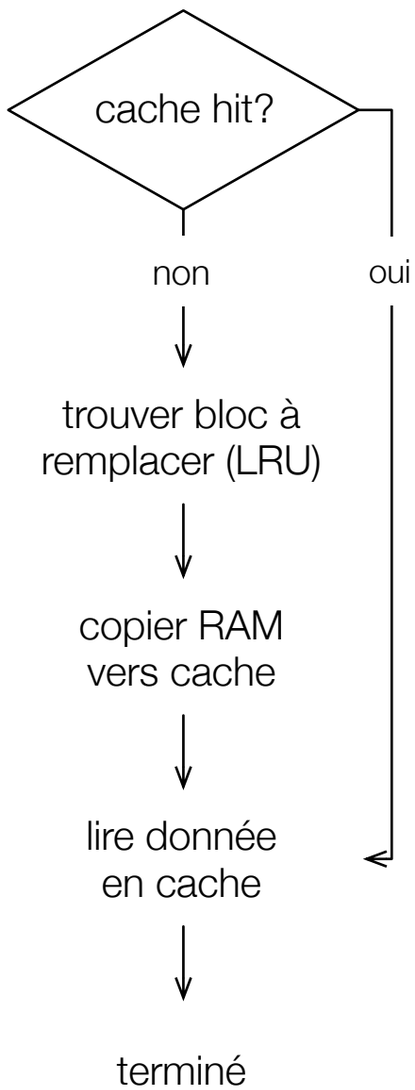


On rajoute une cache!

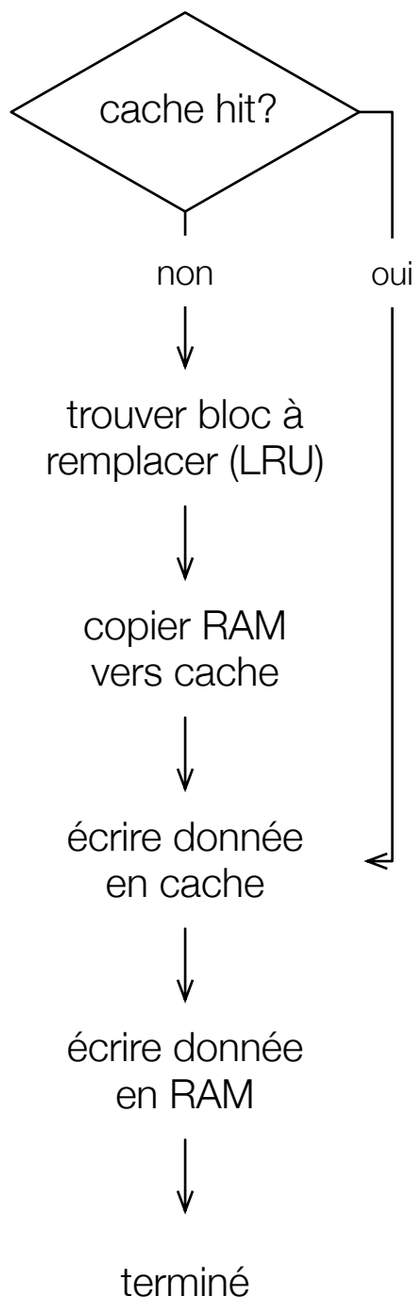


Cache « write-through »

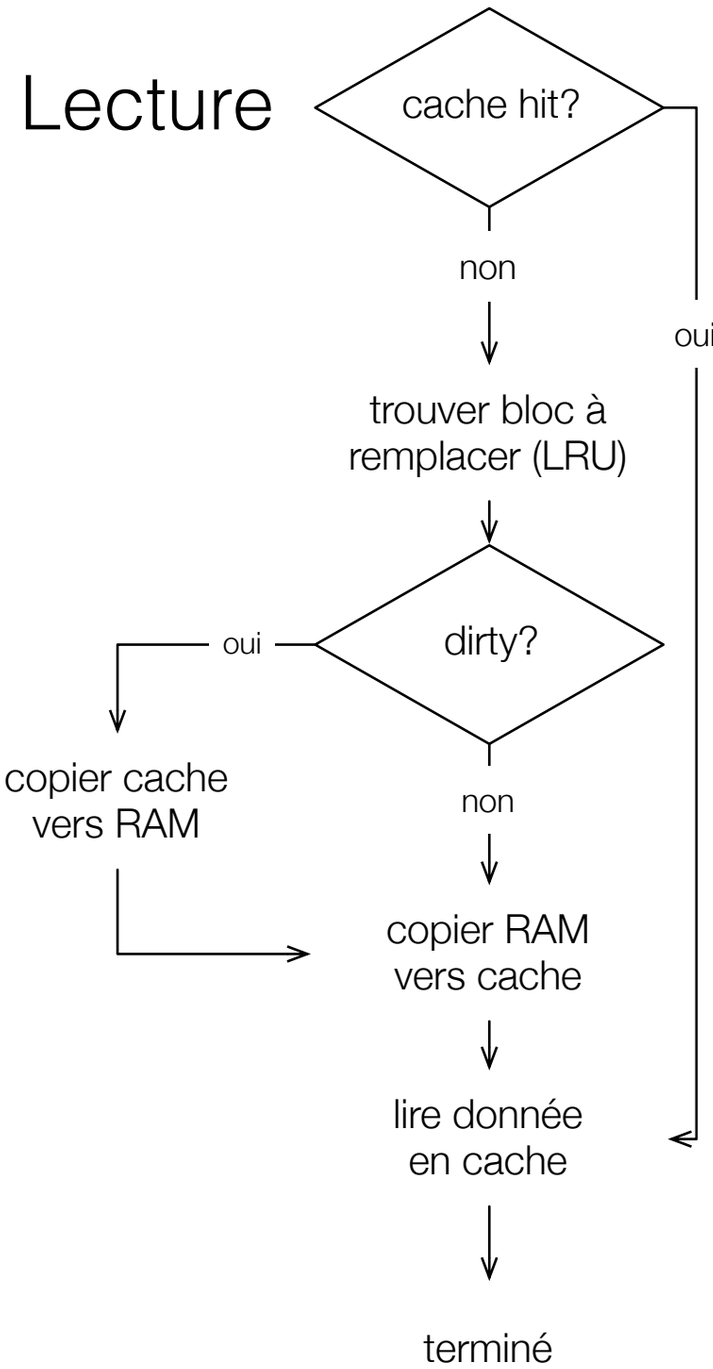
Lecture

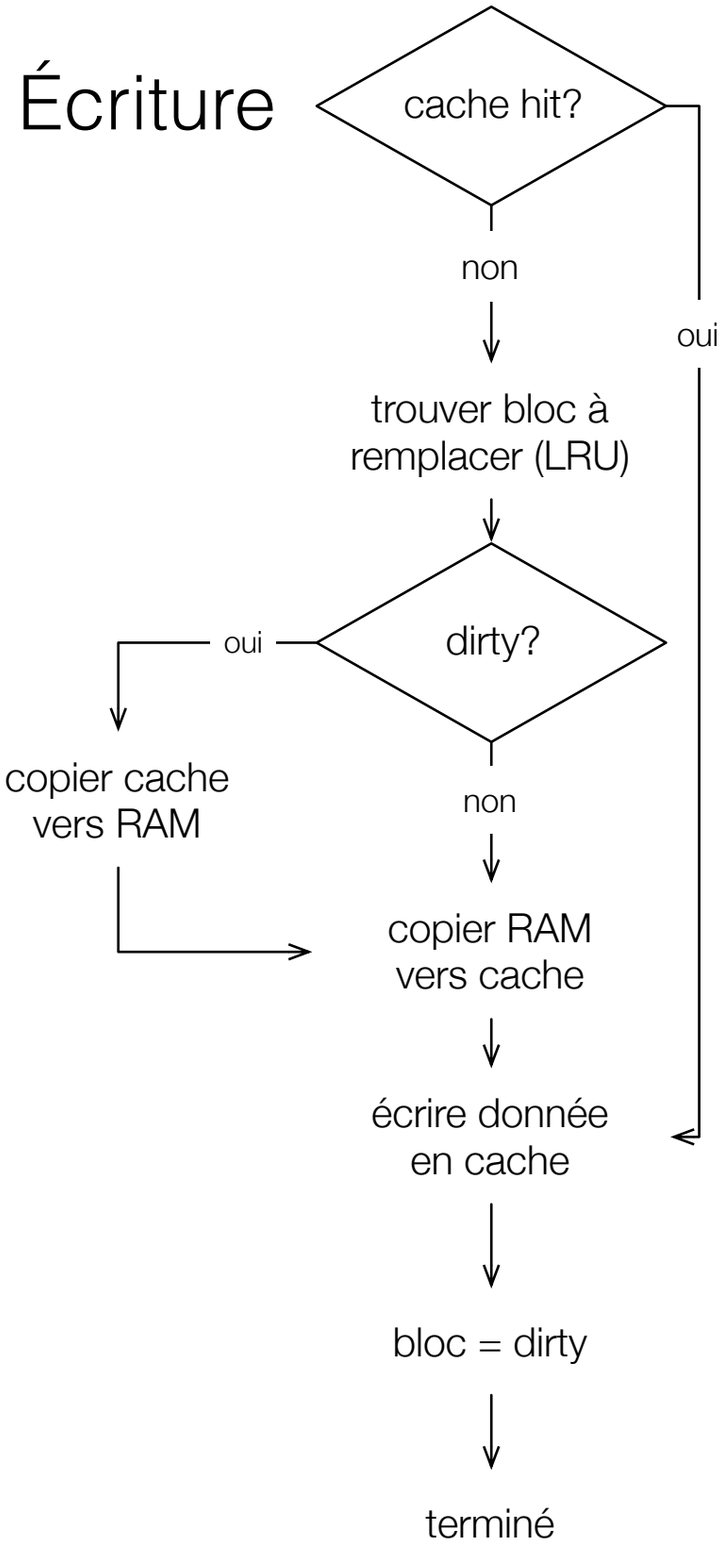


Écriture

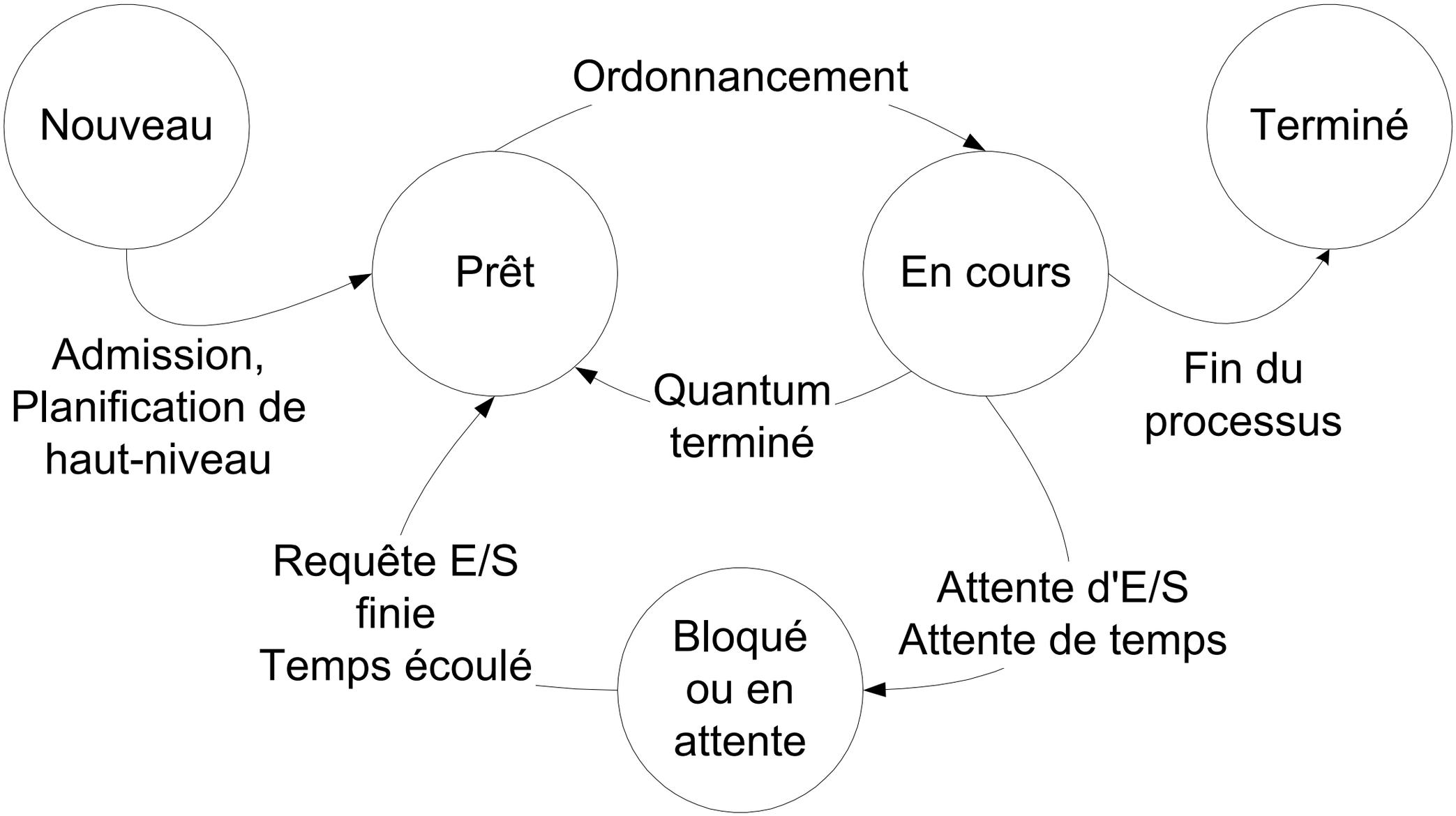


Cache « write-back »





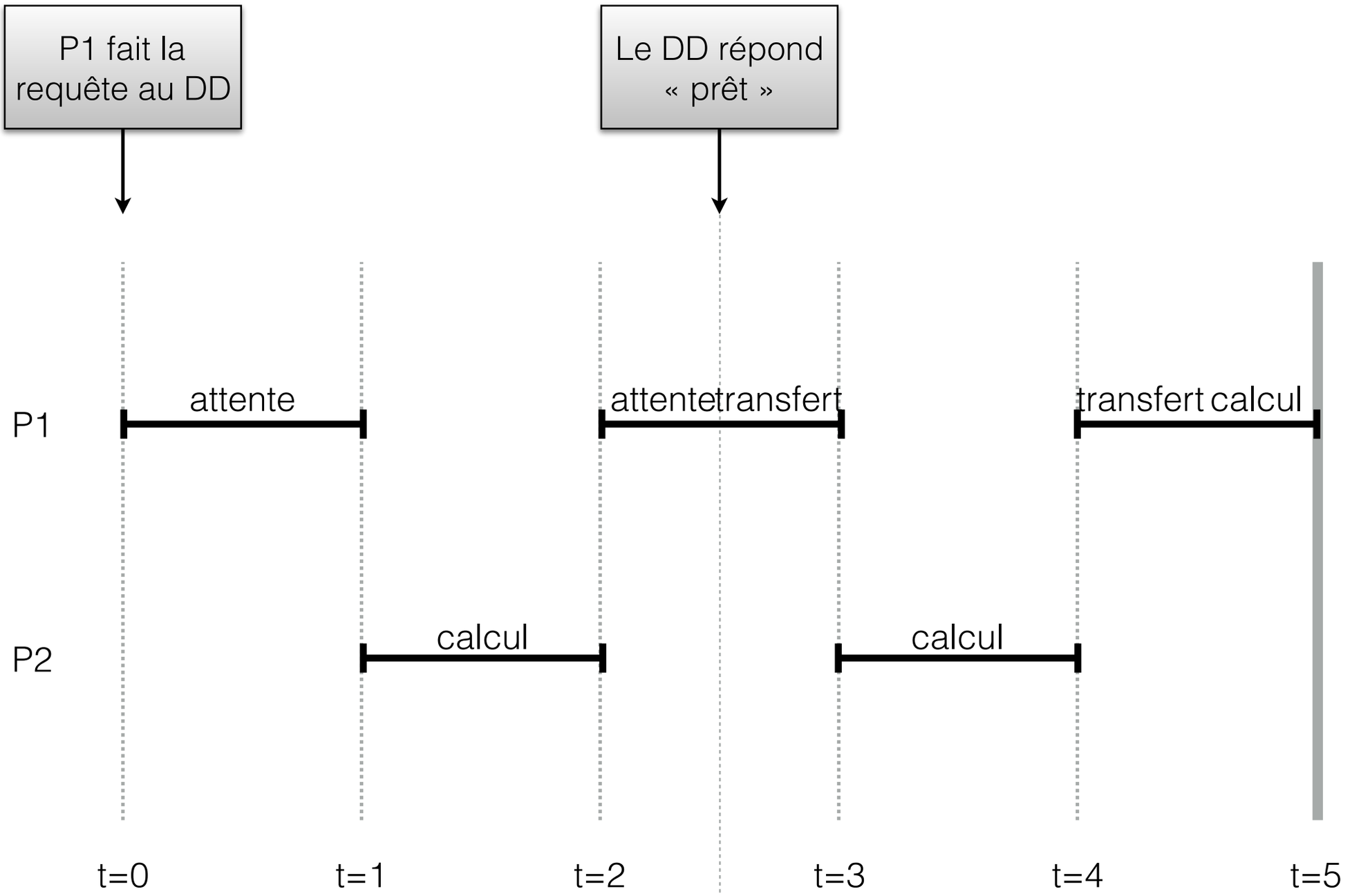
États des processus



Entrées/Sorties

- Il existe trois techniques principales pour communiquer à partir du CPU/Mémoire vers un périphérique à travers un module de I/O:
  - E/S programmées
  - E/S avec interruptions
  - le DMA

Exemple: E/S programmées

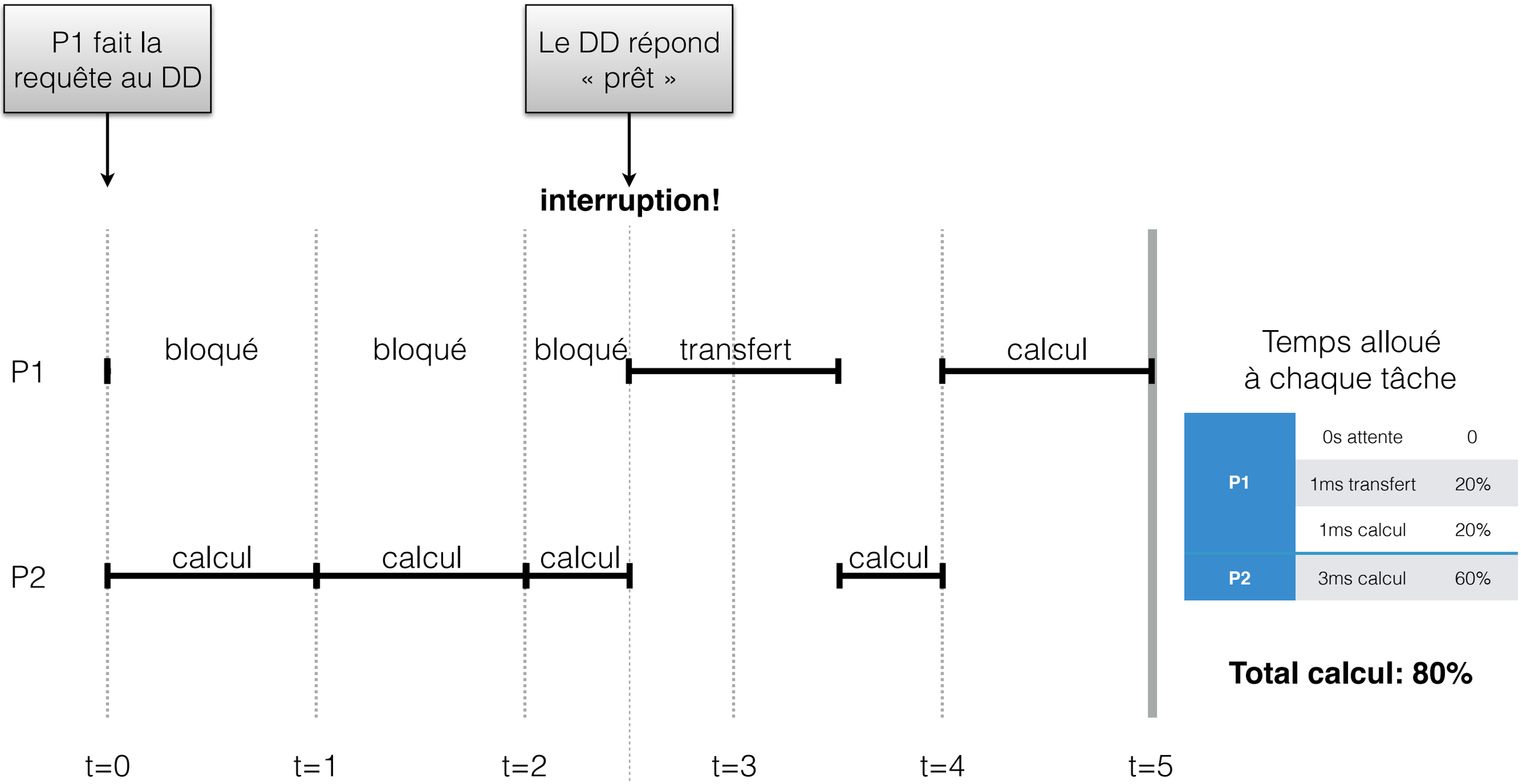


Temps alloué à chaque tâche

P1	1.5ms attente	30%
	1ms transfert	20%
	.5ms calcul	10%
P2	2ms calcul	40%

Total calcul: 50%

Exemple: E/S par interruptions

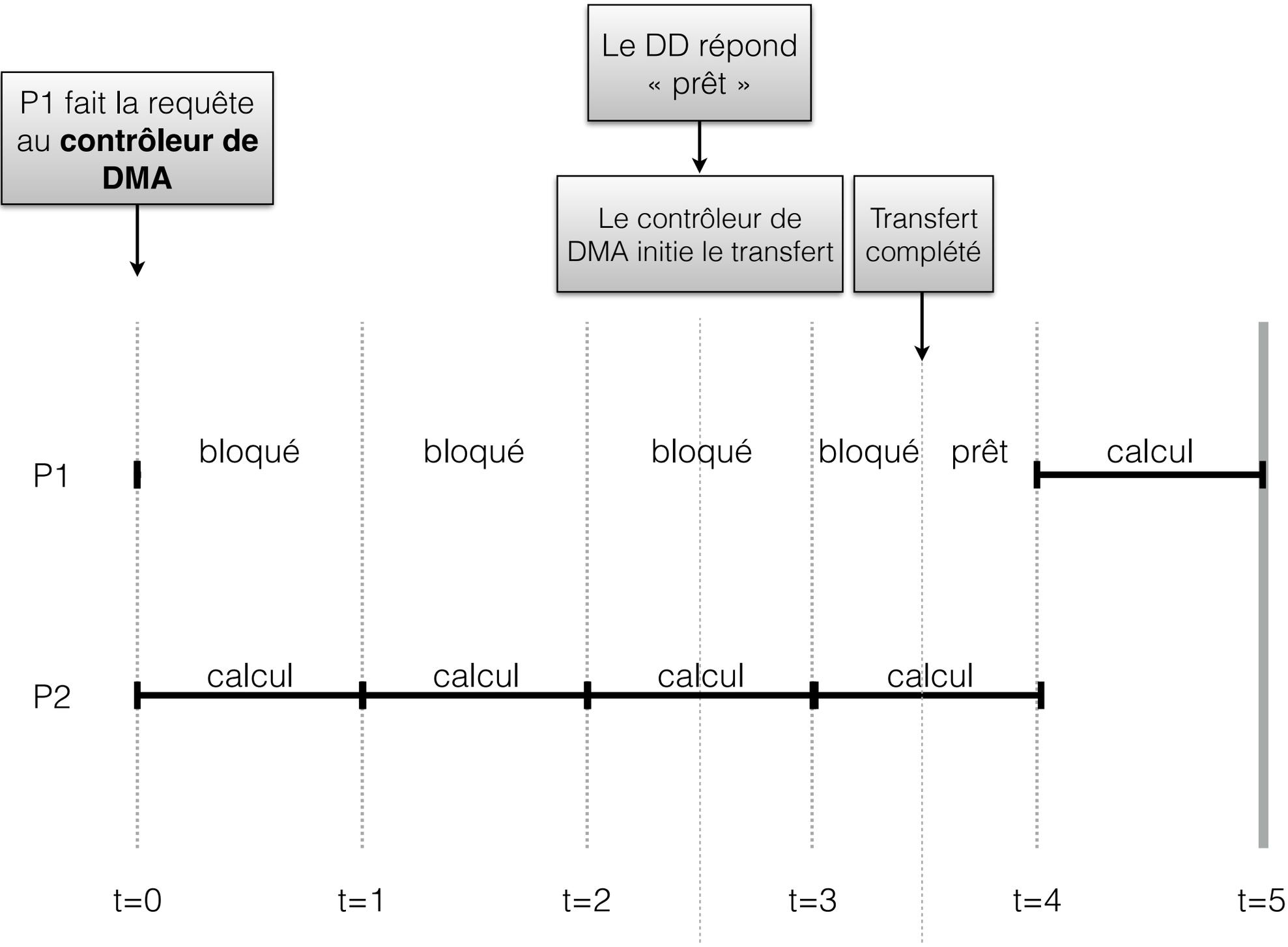


Temps alloué à chaque tâche

P1	0s attente	0
	1ms transfert	20%
	1ms calcul	20%
P2	3ms calcul	60%

Total calcul: 80%

Exemple: E/S par DMA



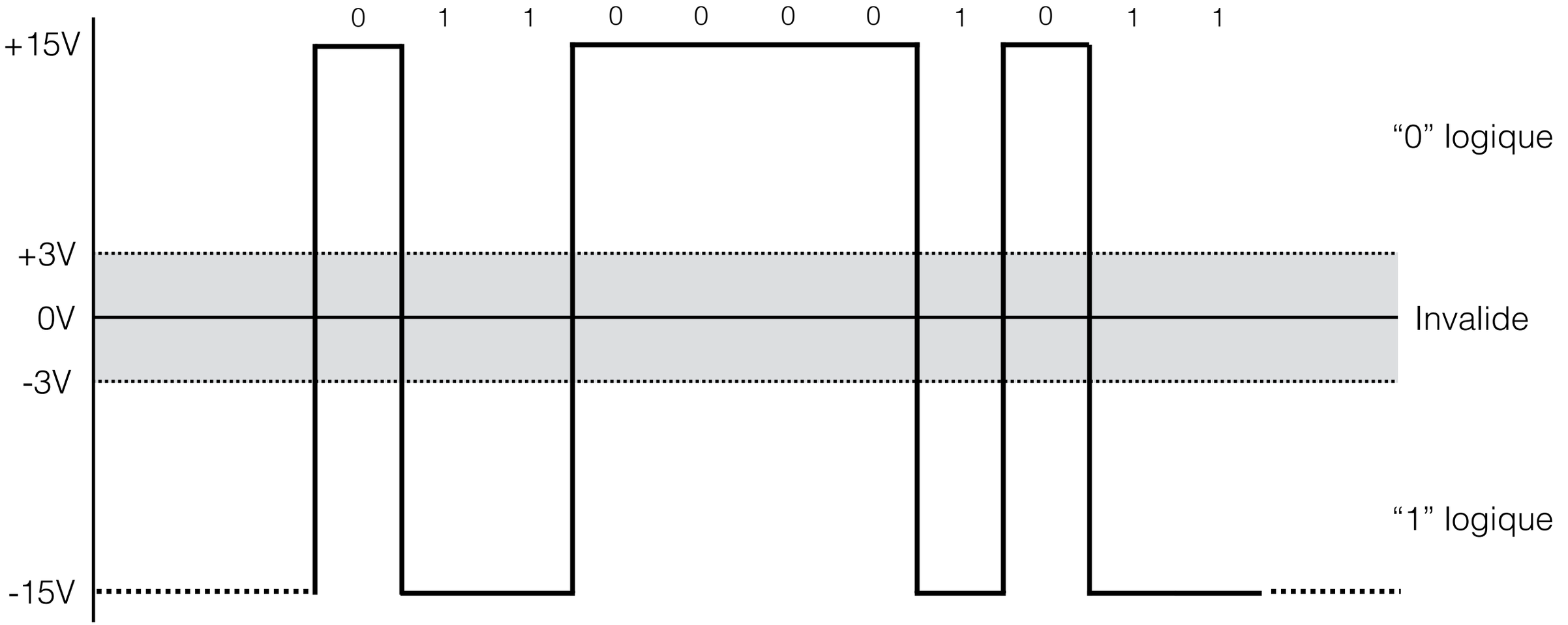
Temps alloué à chaque tâche

P1	0s attente	0%
	1ms transfert	0%
	1ms calcul	20%
P2	4ms calcul	80%

Total calcul: 100%

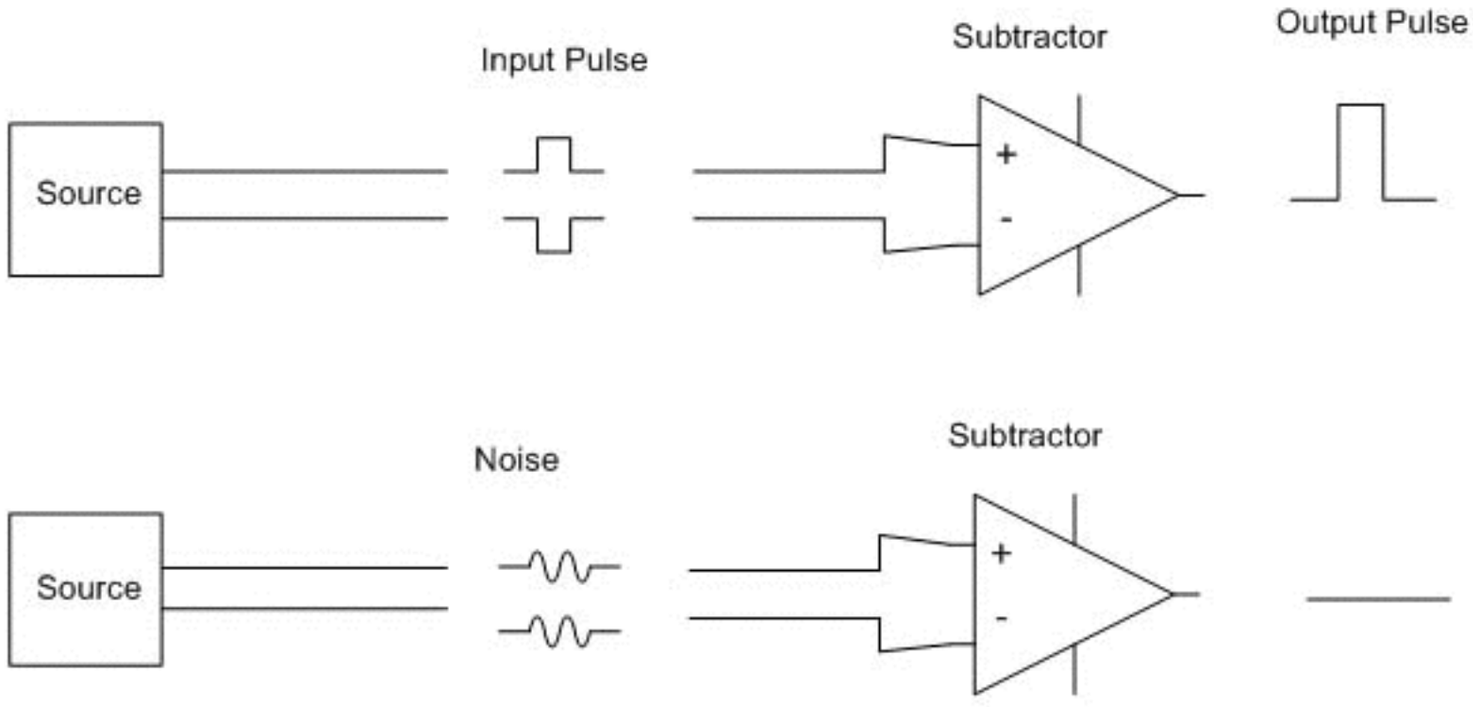
Port série

- On «emballe» le mot à envoyer avec:
  - un bit de départ pour indiquer le début du mot
  - un bit de parité pour détecter les erreurs
  - un bit d'arrêt pour indiquer la fin du mot



Transmission différentielle

- Les bits transmis sont encodés en mode différentiel. Pourquoi?
  - La différence de tension entre deux signaux propagés sur deux lignes différentes détermine la valeur d'un bit transmis. Des symboles différents sont transmis si la différence est positive ou négative.
  - Le bruit commun sur les deux lignes propageant le signal est éliminé lorsque la différence est effectuée. Très robuste.
  - Lorsque la différence est nulle, le bit est invalide ou une autre information peut être transmise.



Topologie d’un réseau USB

- Un réseau USB a une topologie en étoile.
- Le port USB est contrôlé entièrement par un contrôleur unique appelé hôte (“host”). Souvent le PC, il initie toutes les communications, et est le maître absolu du bus.
- Les “hubs” permettent de relier plusieurs appareils à un seul port USB.
  - Le rôle principal des hubs est de transférer les données de l’hôte aux périphériques.
  - Chaque hub contrôle ses ports afin de savoir si un appareil s’y connecte
  - Il peut y avoir 5 niveaux de hub en plus du hub racine.
- Il y a 127 appareils maximum dans un réseau USB. Chaque appareil a son adresse.