Architecture systèmes Numériques

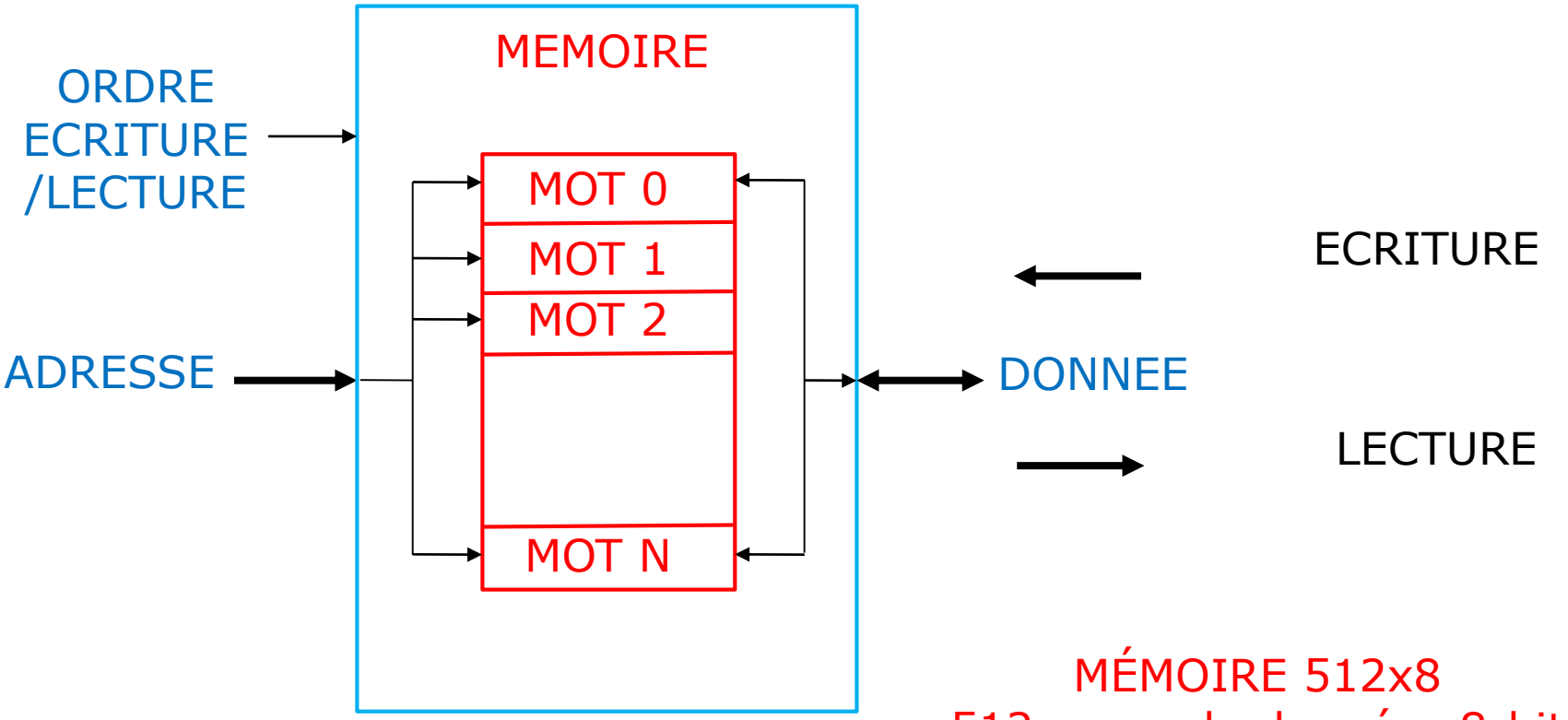
Hervé Le Provost

DSM/IRFU - CEA Saclay
herve.le-provost@cea.fr

avec la participation de Shebli Anvar
shebli.anvar@cea.fr

Version 2.0 – 2 octobre 2017

Circuits numériques : la mémoire



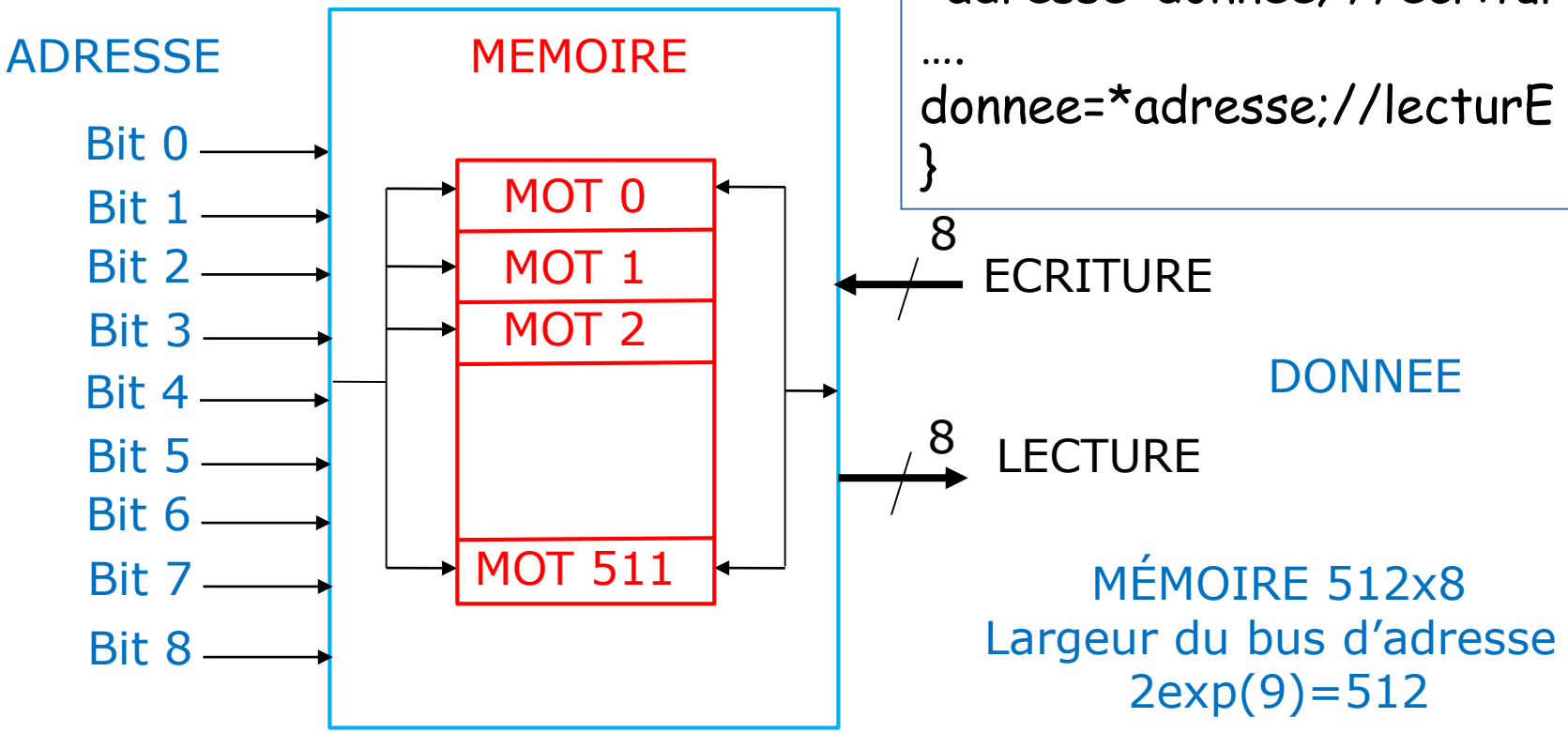
MÉMOIRE 512x8
512 cases de données 8-bit
Largeur du bus d'adresse ?

herve.le-provost@cea.fr

Circuits numériques : la mémoire

ECRITURE à l'adresse 48 de la valeur 6 / LECTURE

```
int main() {
char *adresse;
char donnee=6;
adresse=(char*)0x30;
*adresse=donnee; //ecriture
....
donnee=*adresse;//lecturE
}
```



MÉMOIRE 512x8
Largeur du bus d'adresse
2exp(9)=512

herve.le-provost@cea.fr

Circuits numériques : la mémoire

- Mémoires non Volatiles
  - ROM (Read Only Memory), programmée à la fabrication
  - PROM (Programmable ROM), programmée une seule fois par l'utilisateur.
  - EPROM (Erasable PROM), programmable plusieurs fois par l'utilisateur sur des machines dédiées
  - EEPROM (Electrically Erasable PROM), programmable plusieurs fois par l'utilisateur in-situ
    - Les plus utilisées : FLASH, BIOS carte mère PC. Rapides et haute densité (plusieurs Méga octets)
  - NVRAM (Non volatile Random Access Memory)
- Mémoires Volatiles
  - RAM (Random Access Memory), SRAM, SDRAM, DDR2, DDR3.

herve.le-provost@cea.fr

Circuits numériques : codage des valeurs

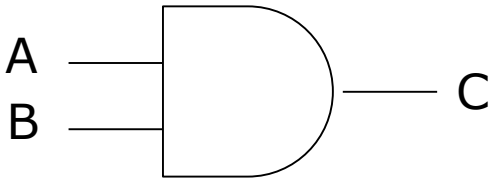
décimal	binaire	Hexa-décimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7

décimal	binaire	Hexa-décimal
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Valeur décimale 47, codage binaire et hexadécimal ?

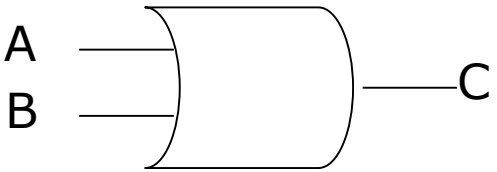
herve.le-provost@cea.fr

Circuits numériques : portes logiques



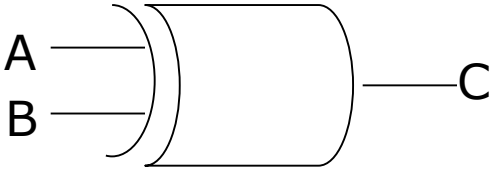
ET

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1



OU

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

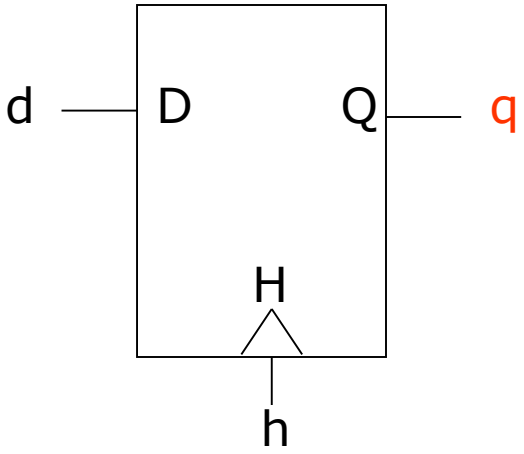


OU EXCLUSIF

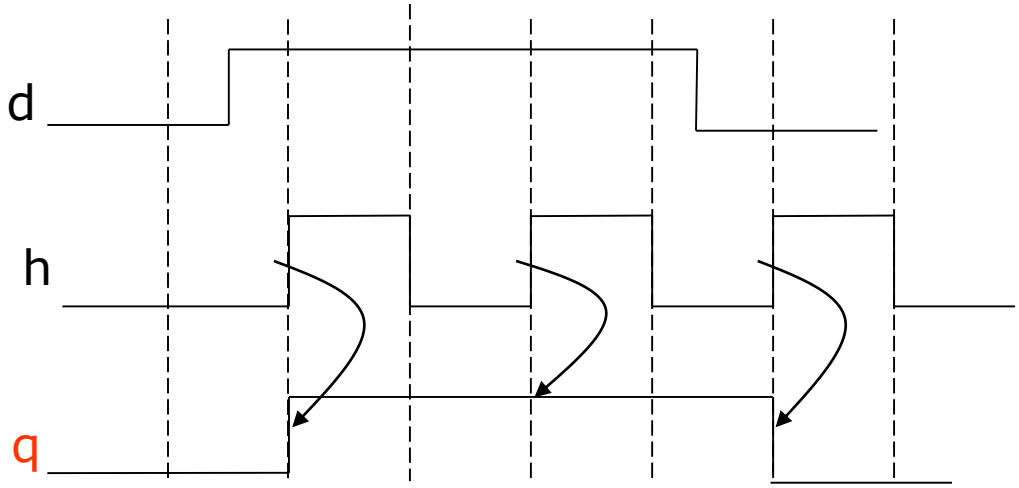
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

herve.le-provost@cea.fr

Circuits numériques : Bascule D



Bascule D



Chronogramme

h	D	Q_{n+1}
↑	0	0
↑	1	1
↓	x	Q_n
0	x	Q_n
1	x	Q_n

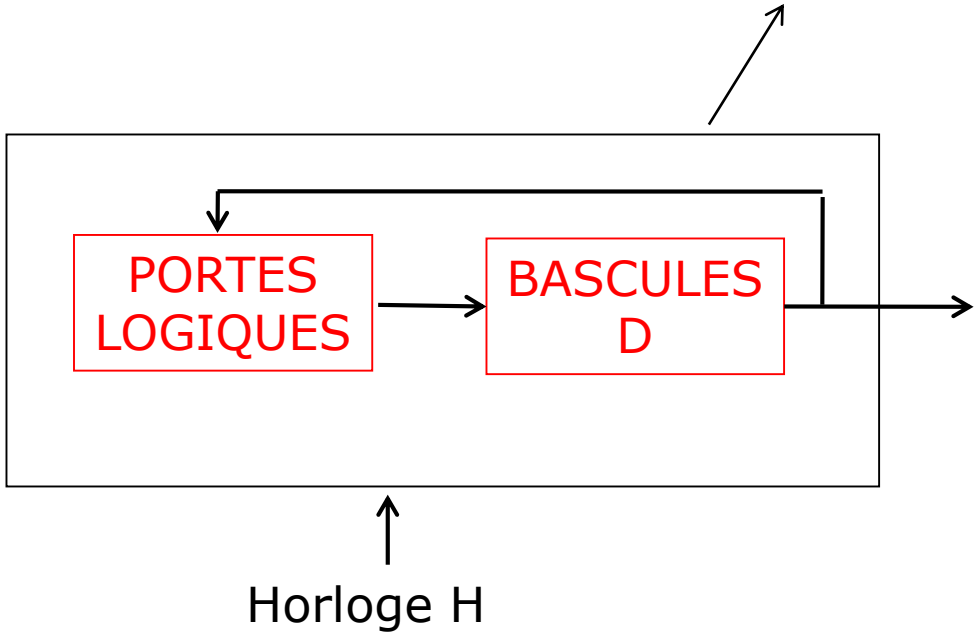
Table de vérité

herve.le-provost@cea.fr

Circuits numériques : Design Synchrone

EXEMPLE : UN COMPTEUR

FPGA :
FIELD PROGRAMMABLE GATE ARRAY



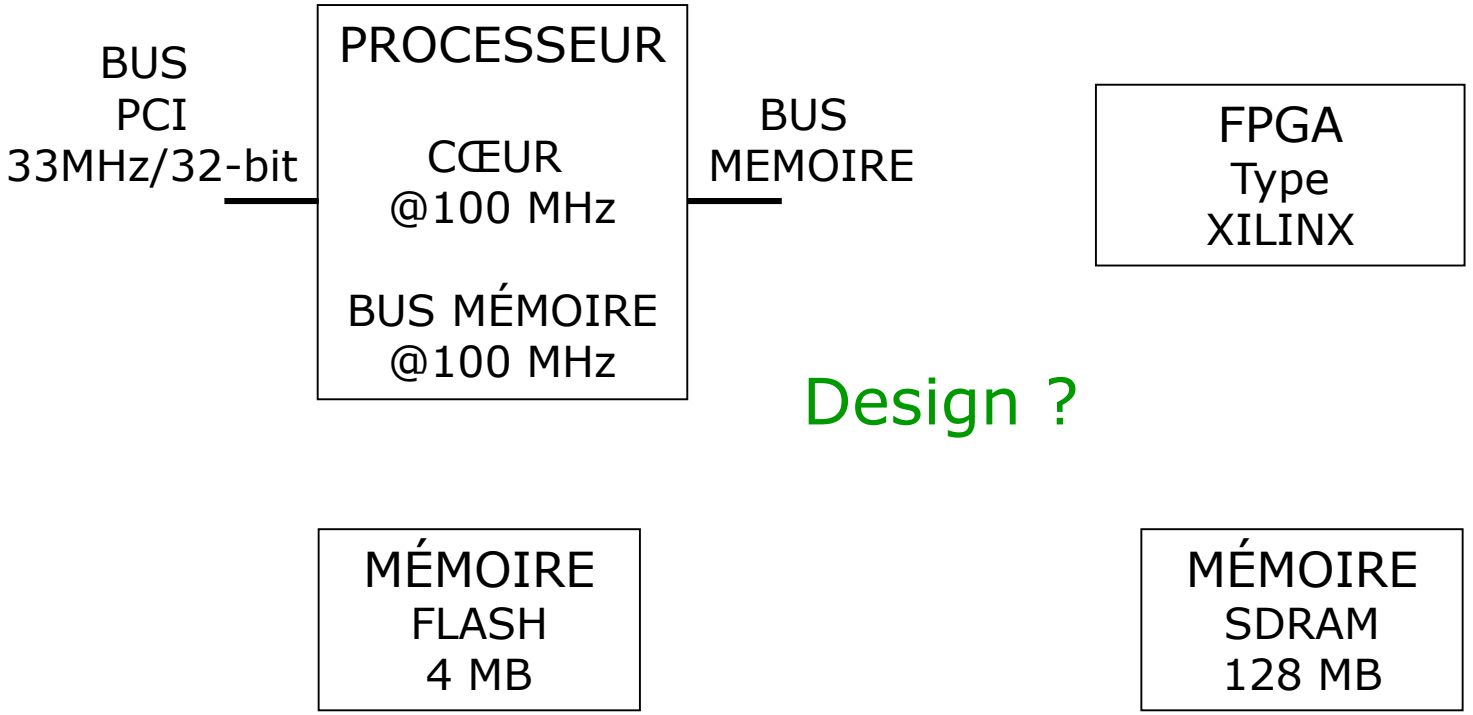
SORTIE
S=S+1
EVOLUTION SUR CHAQUE FRONT MONTANT DE L'HORLOGE H

FREQUENCE F=50 MHz, COMPTEUR S CODE SUR 8 BIT => QUELLE EST LA DUREE ECOULEE ENTRE 2 PASSAGE A 0 DU COMPTEUR ?

herve.le-provost@cea.fr

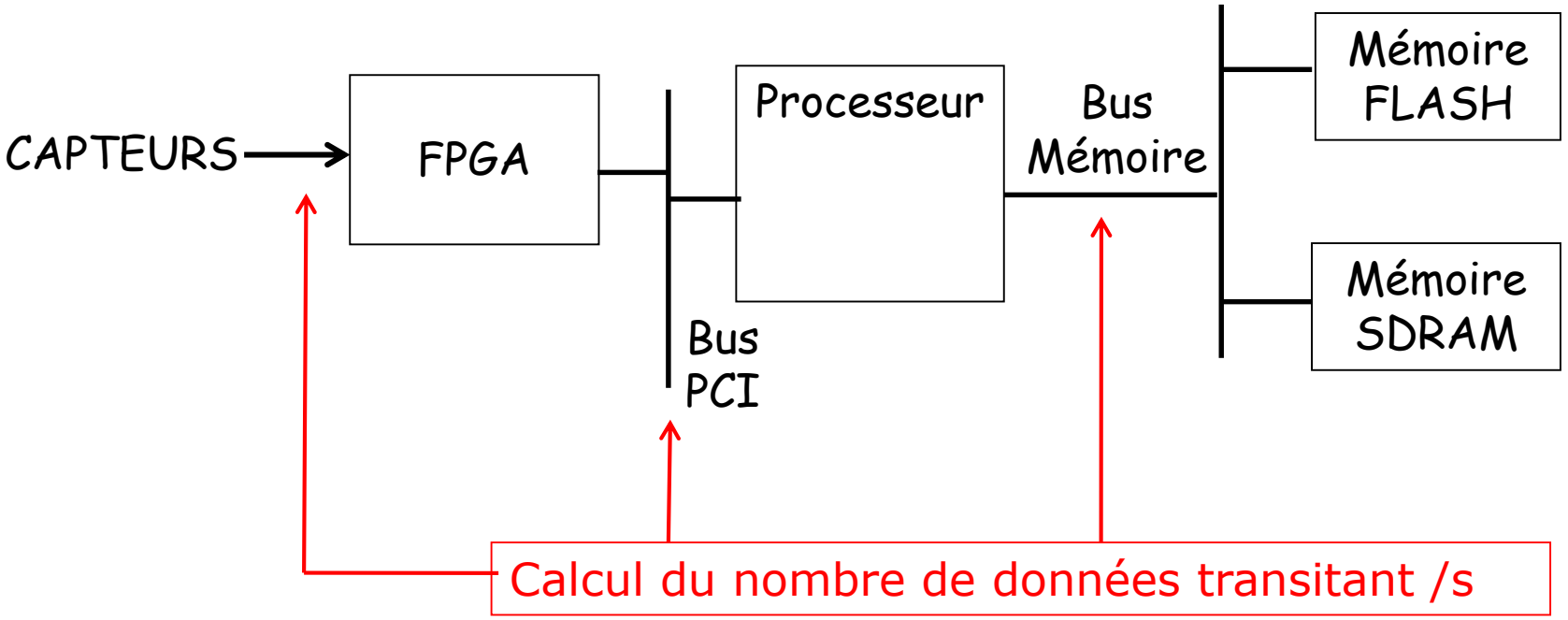
Circuits numériques : un design, un LEGO

DESIGN : LECTURE DE CAPTEURS (FORMAT PROPRIETAIRE)
ET TRAITEMENT DES DONNEES PAR UN PROCESSEUR
TOURNANT UN SYSTÈME D'EXPLOITATION



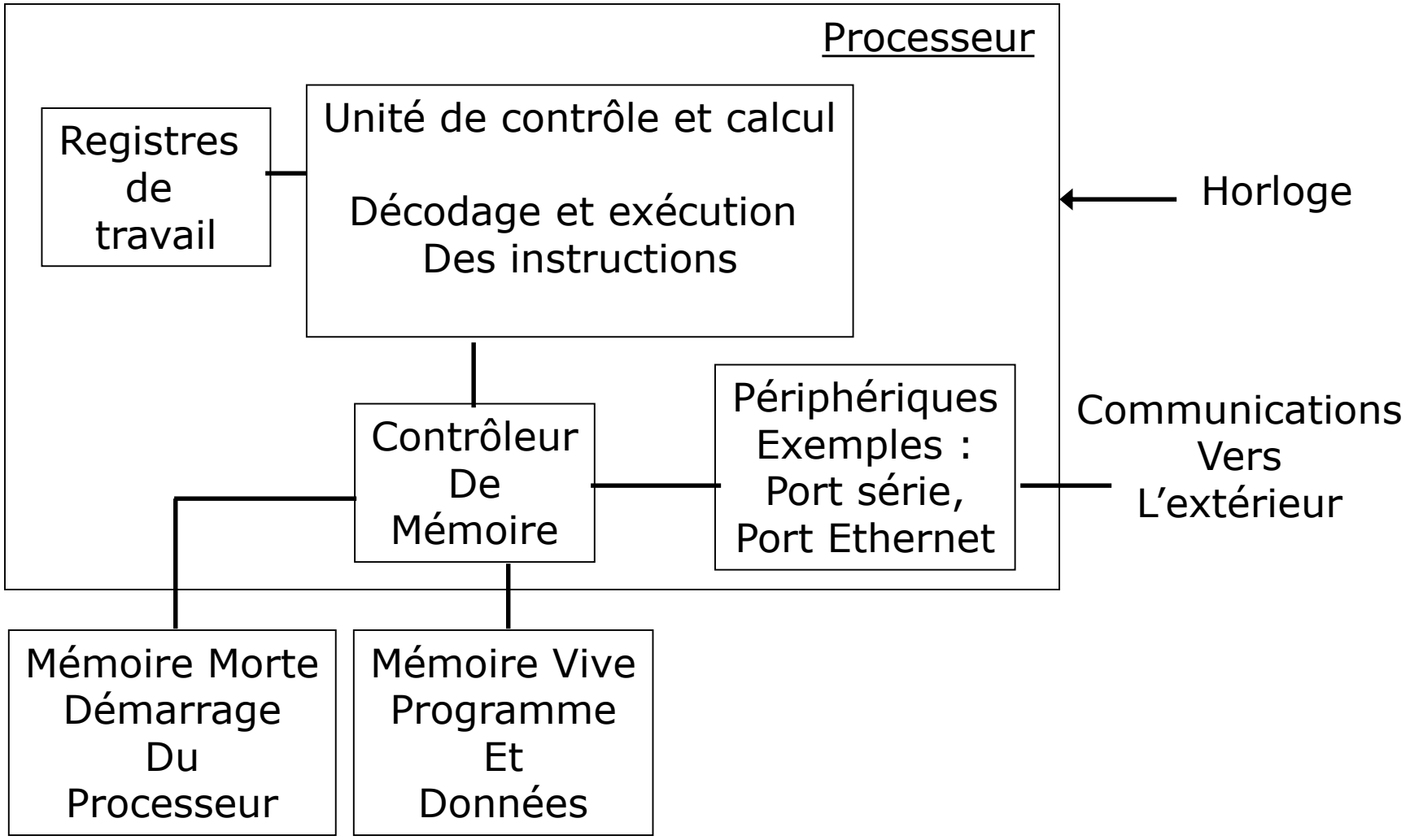
herve.le-provost@cea.fr

Circuits numériques : un design, un LEGO



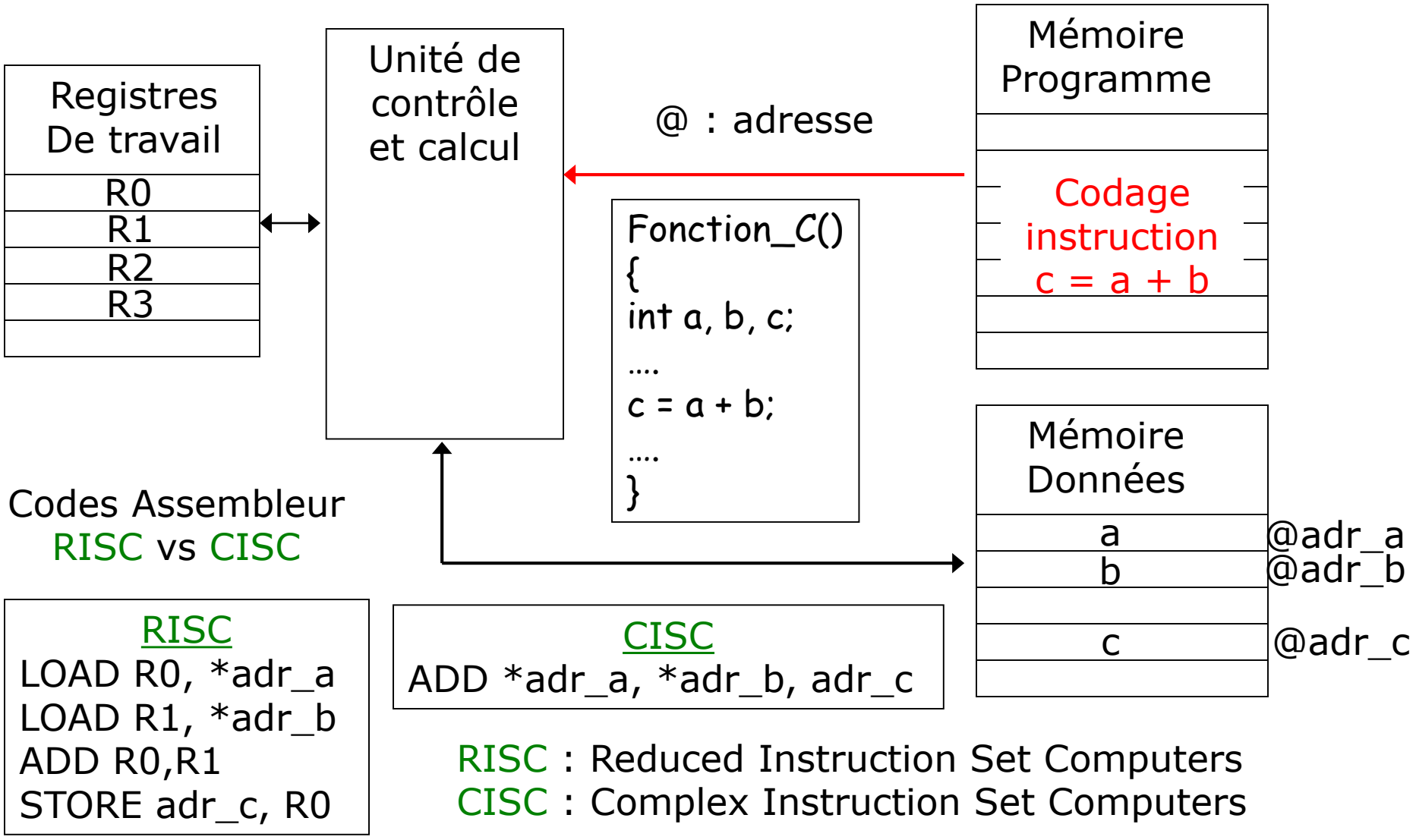
herve.le-provost@cea.fr

Architecture générique d'une carte processeur



herve.le-provost@cea.fr

Processeur et codage assembleur



Codes Assembleur
RISC vs CISC

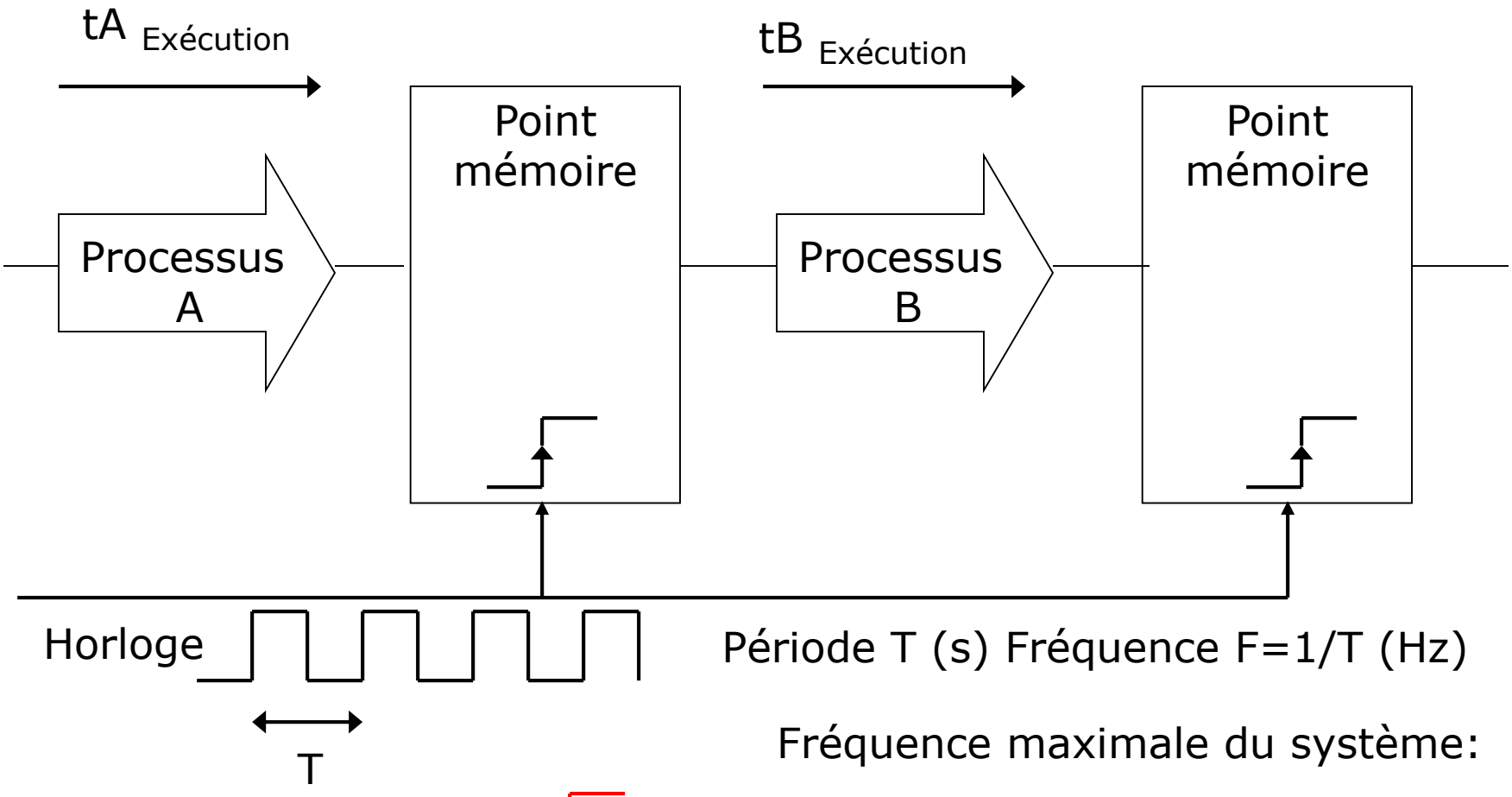
RISC
LOAD R0, *adr_a
LOAD R1, *adr_b
ADD R0,R1
STORE adr_c, R0

CISC
ADD *adr_a, *adr_b, adr_c

RISC : Reduced Instruction Set Computers
CISC : Complex Instruction Set Computers

herve.le-provost@cea.fr

Fréquence maximale de fonctionnement



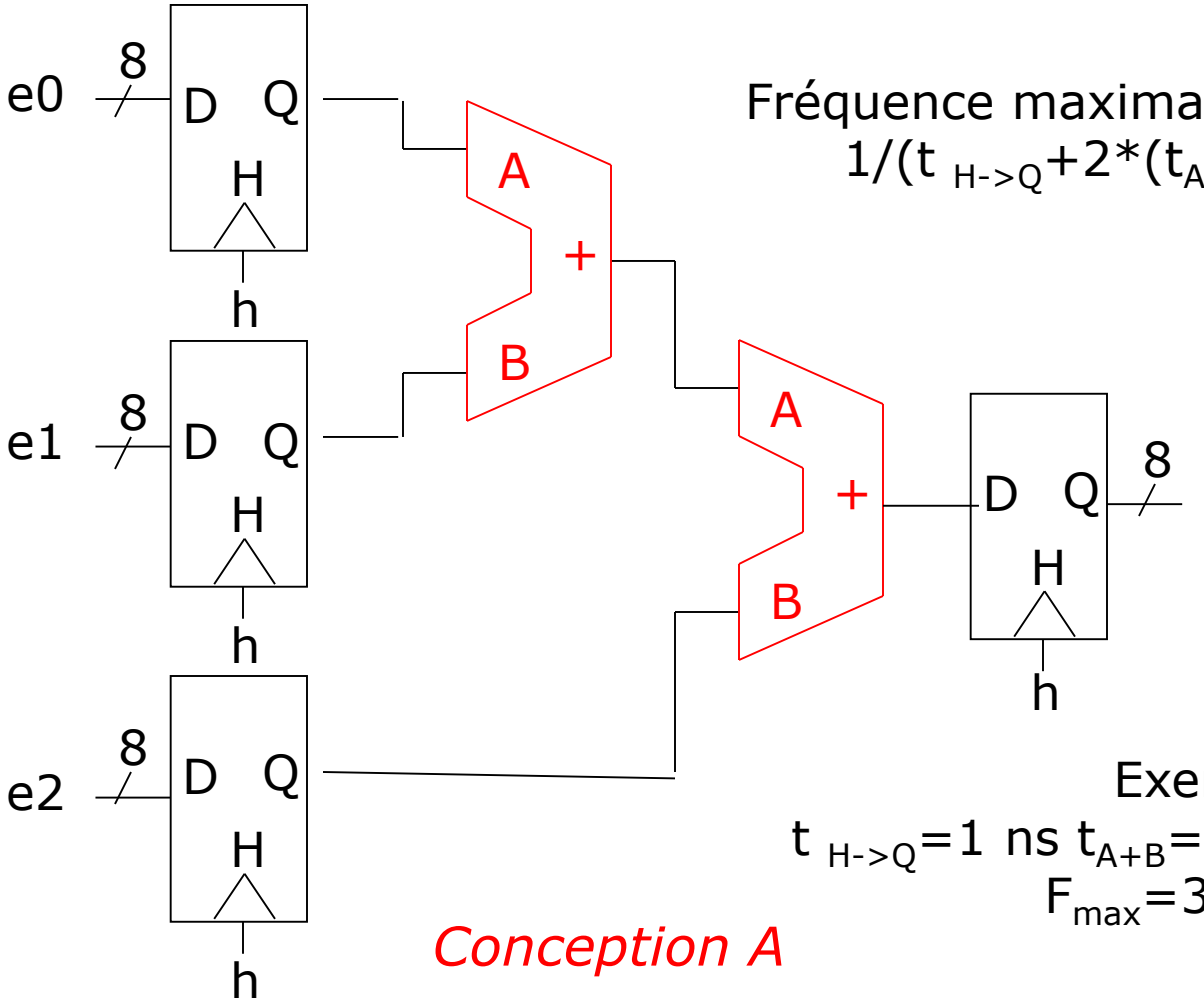
Période T (s) Fréquence F=1/T (Hz)

Fréquence maximale du système:

F_{max}=1/max(tA Exécution, tB Exécution)

herve.le-provost@cea.fr

Fréquence maximale de fonctionnement



Fréquence maximale du système :

$1/(t_{H->Q}+2*(t_{A+B})+t_{D-setup})$

Exemple:

$t_{H->Q}$=1 ns t_{A+B}=15 ns $t_{D-setup}$= 1 ns

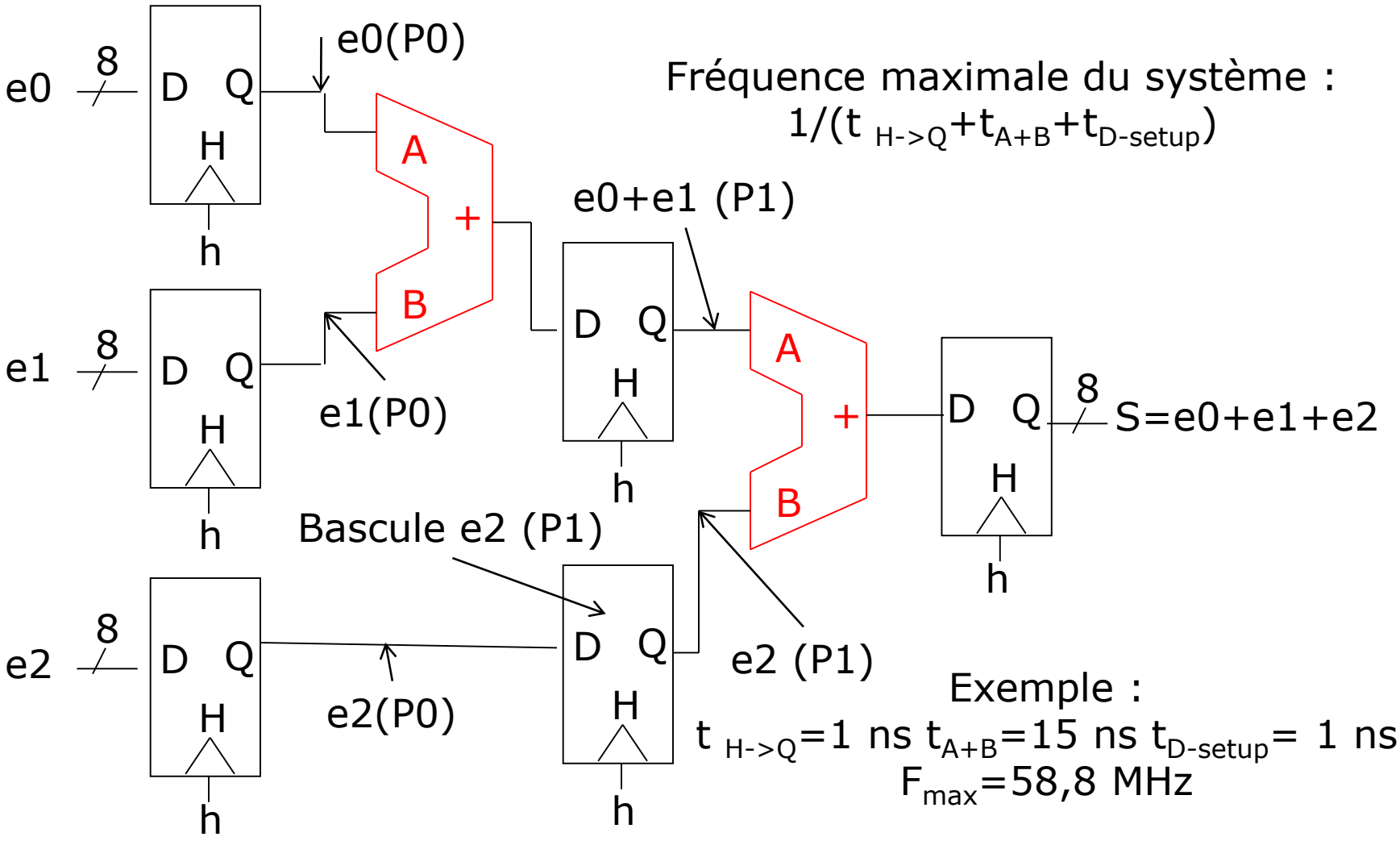
F_{max}=31,2 MHz

Conception A

Proposer une solution pour augmenter la fréquence maximale de fonctionnement

herve.le-provost@cea.fr

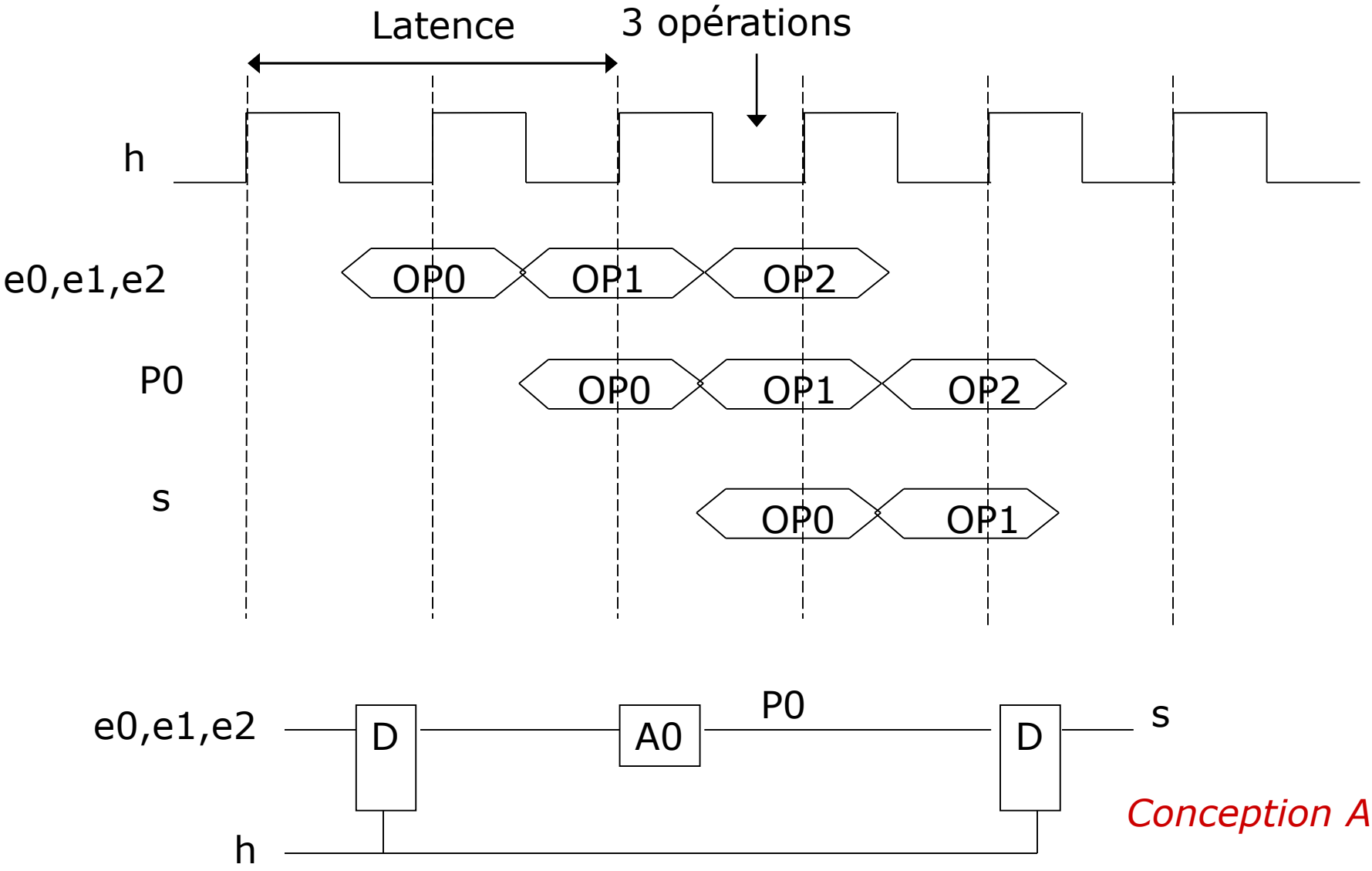
Fréquence maximale de fonctionnement



Conception B: ajout d'un étage de pipeline

herve.le-provost@cea.fr

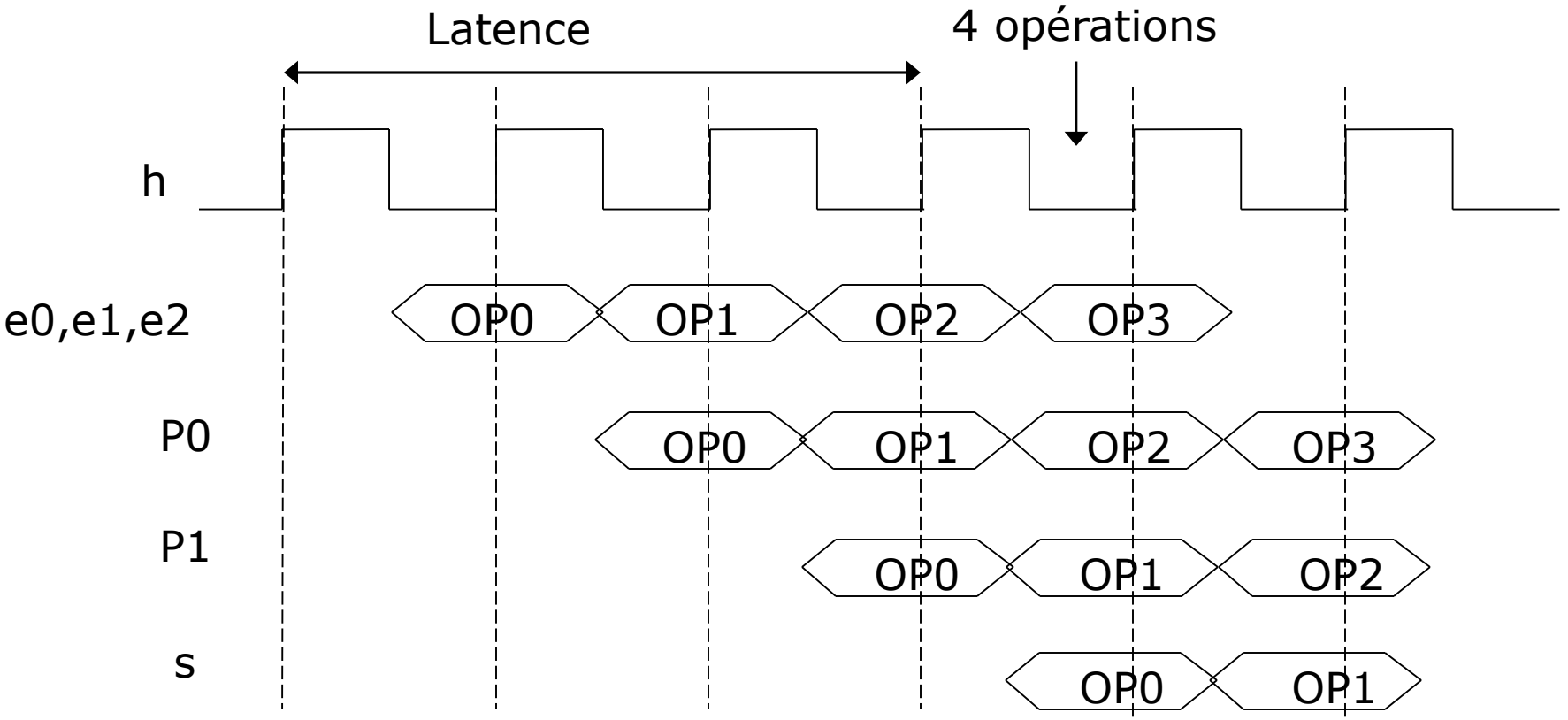
Fréquence maximale de fonctionnement

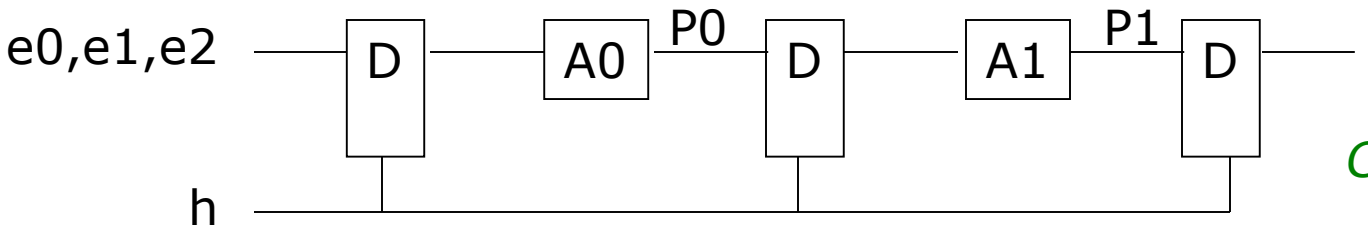


Conception A

herve.le-provost@cea.fr

Fréquence maximale de fonctionnement



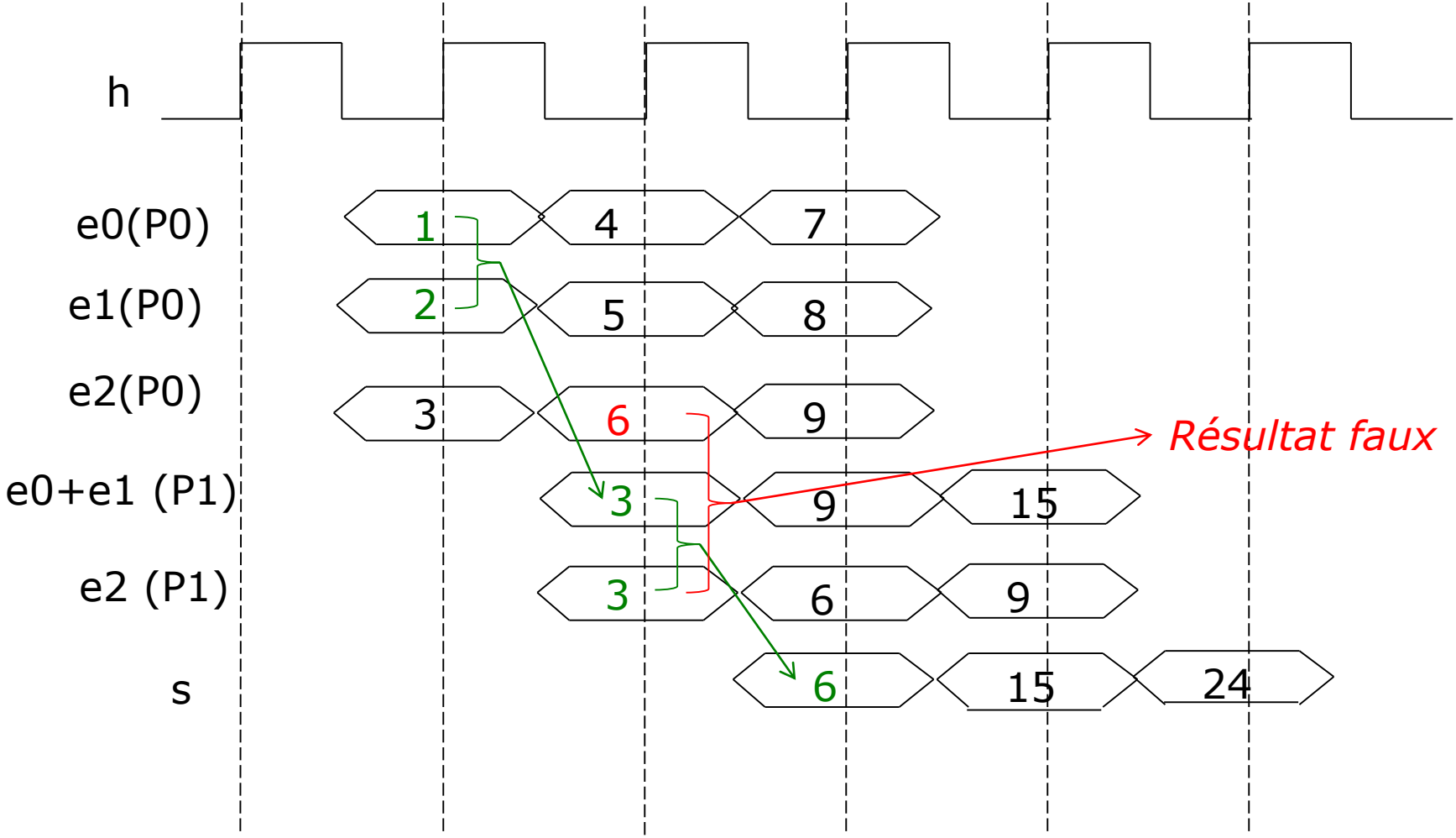


Conception B

herve.le-provost@cea.fr

Fréquence maximale de fonctionnement

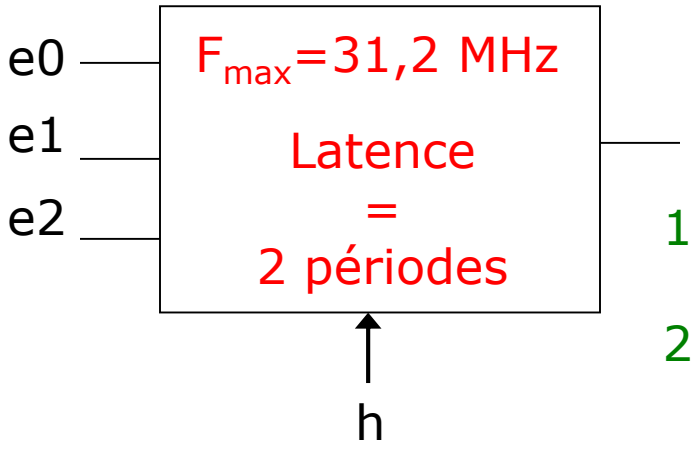
Conception B : de l'utilité de la bascule e2 (P1)



herve.le-provost@cea.fr

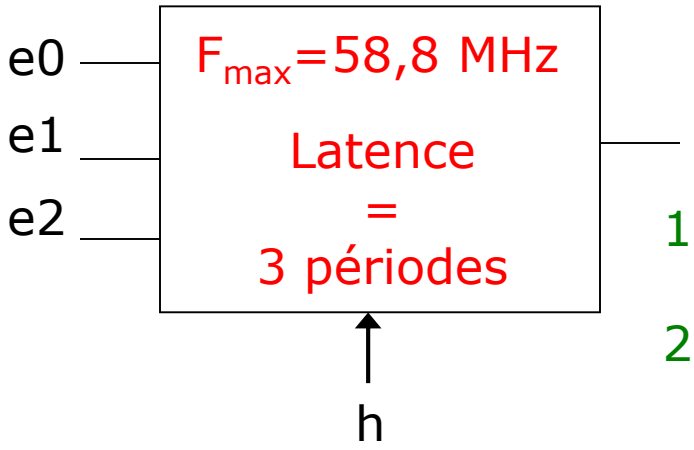
Fréquence maximale de fonctionnement

Conception A



1. Des opérandes toutes les 32 ns. Un résultat toutes les 32 ns
2. Le temps de calcul d'une opération est de 2x32ns=64ns

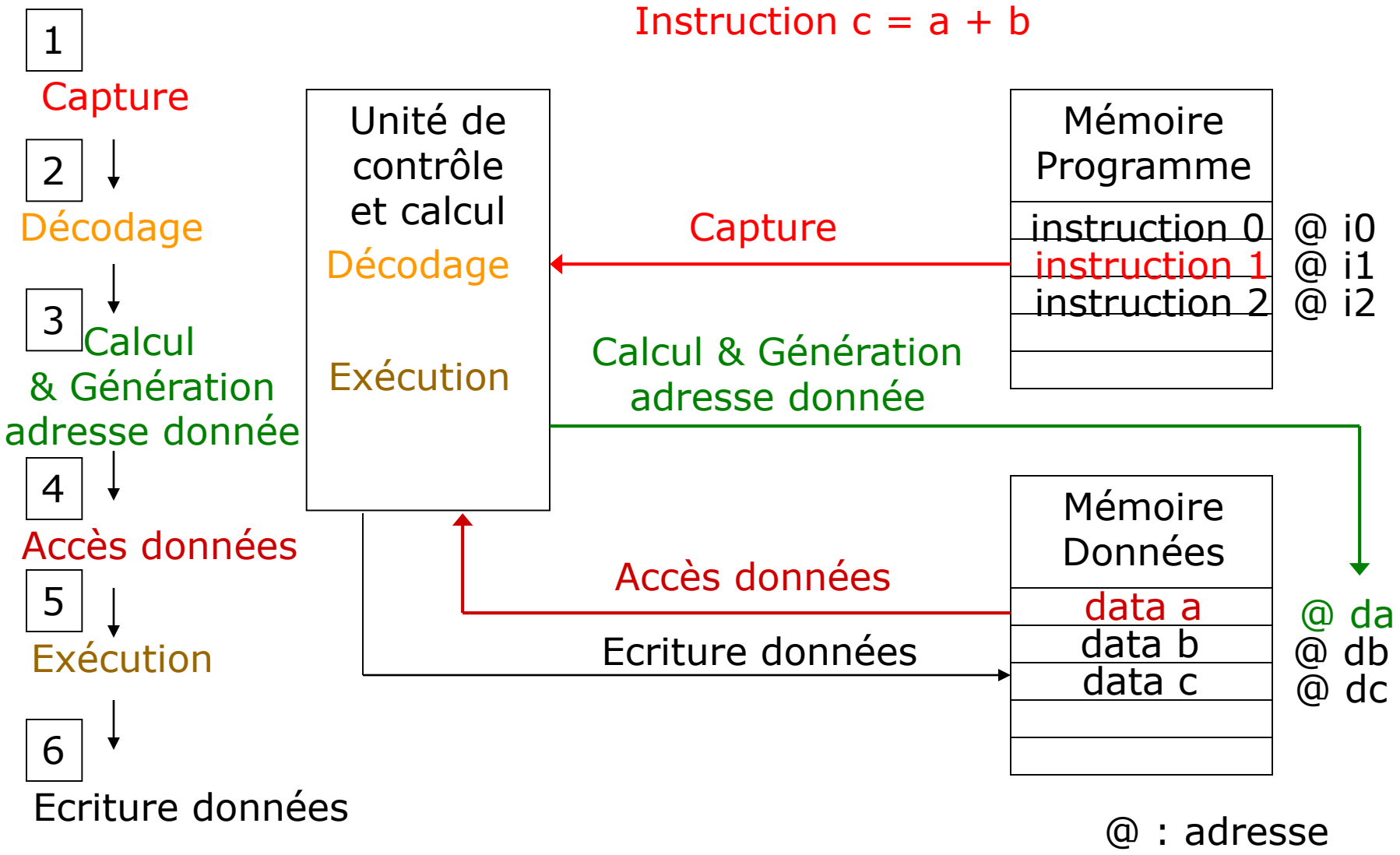
Conception B



1. Des opérandes toutes les 17 ns. Un résultat toutes les 17 ns
2. Le temps de calcul d'une opération est de 3x17ns=51ns

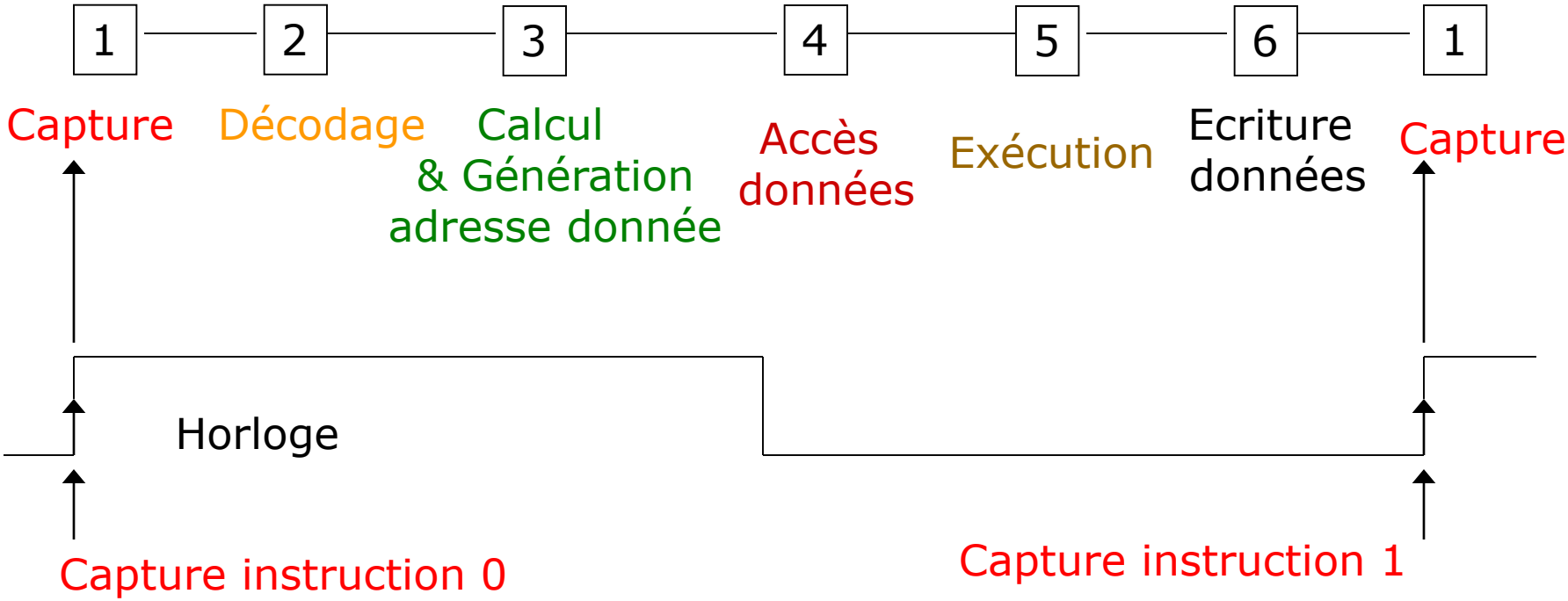
herve.le-provost@cea.fr

Exécution d'une instruction



herve.le-provost@cea.fr

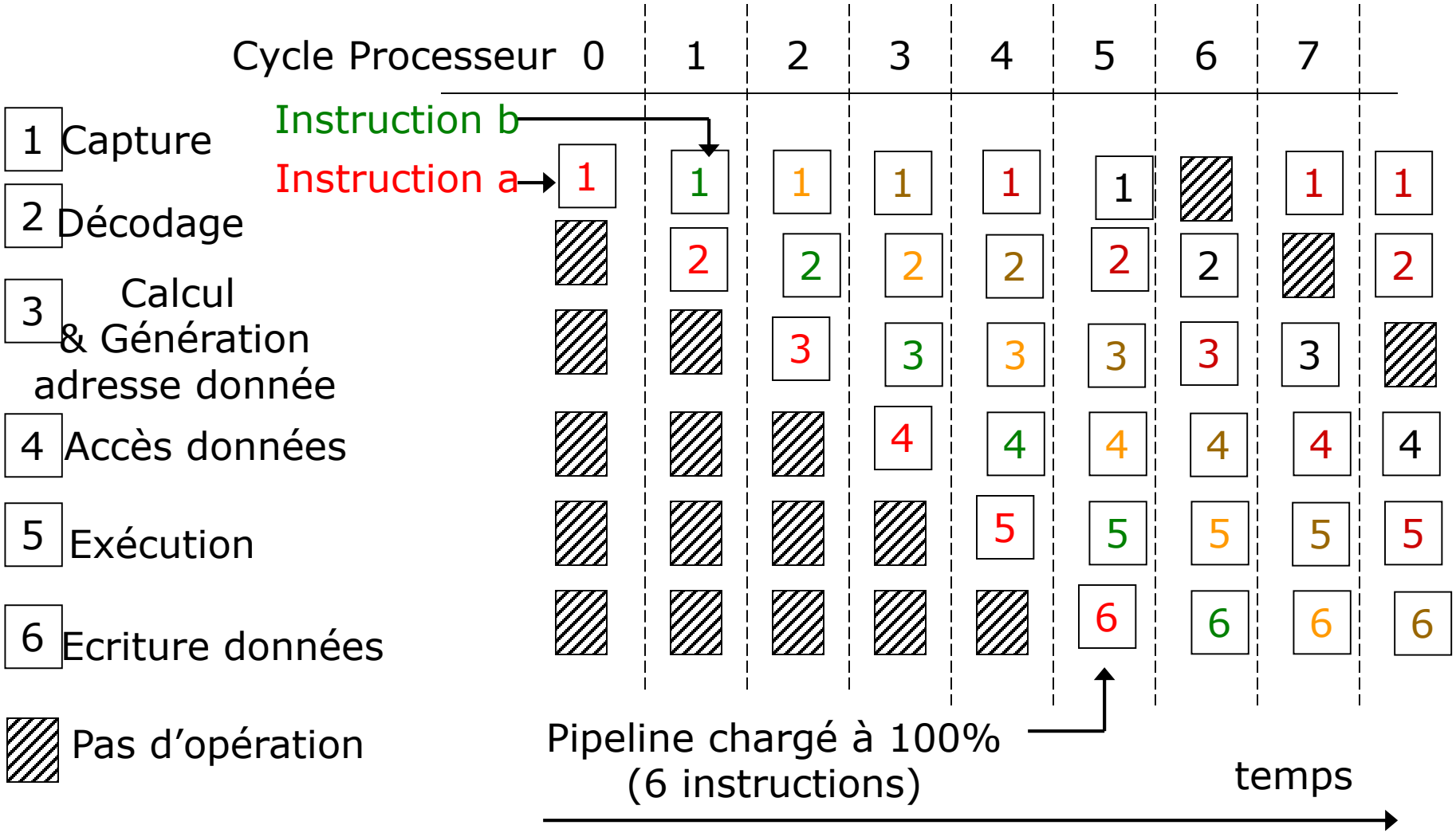
Exécution séquentielle des instructions



Comment augmenter la fréquence maximale de capture des instructions ?

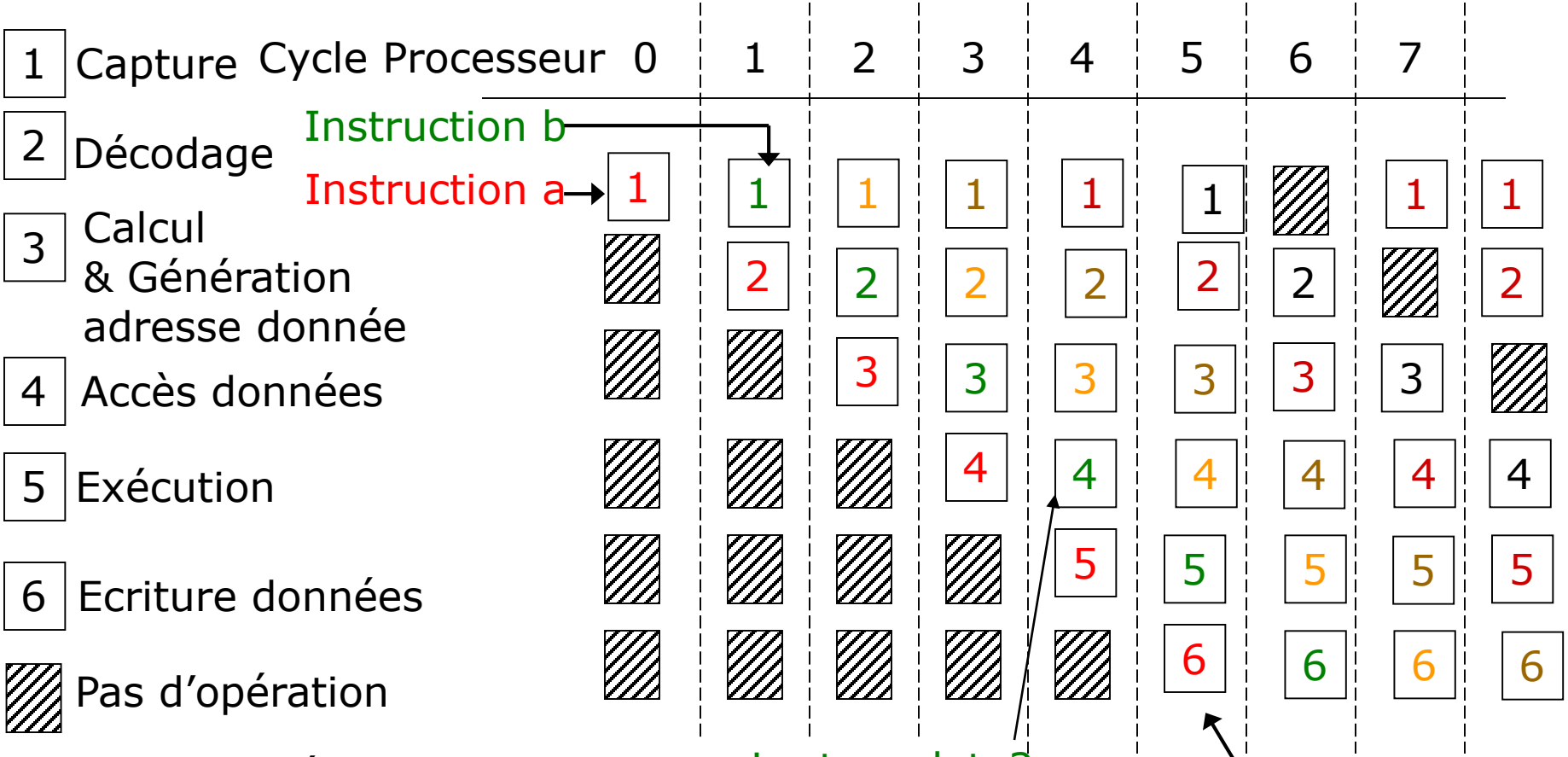
herve.le-provost@cea.fr

Exécution en pipeline des instructions - Principe



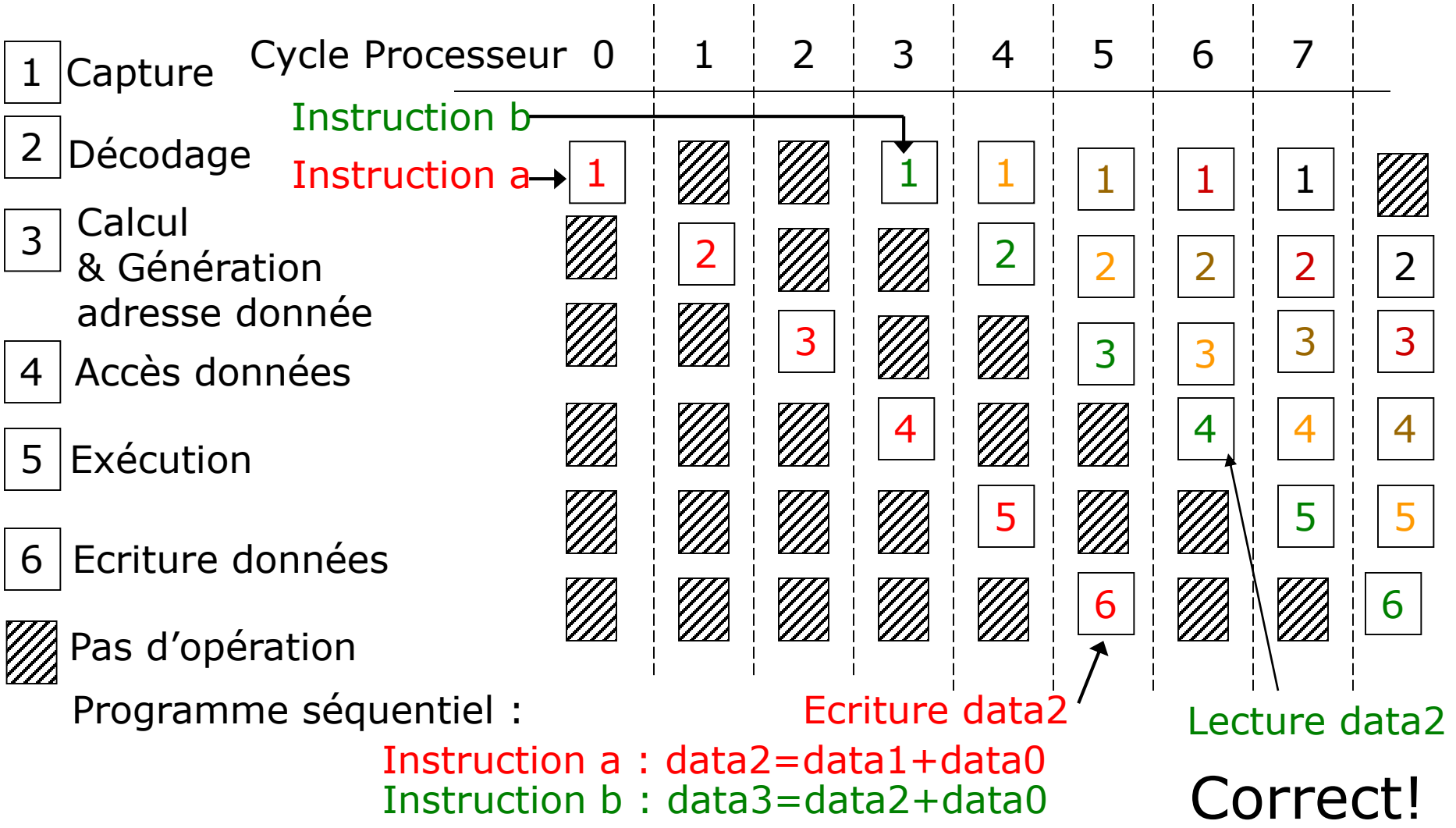
herve.le-provost@cea.fr

Exécution en pipeline des instructions



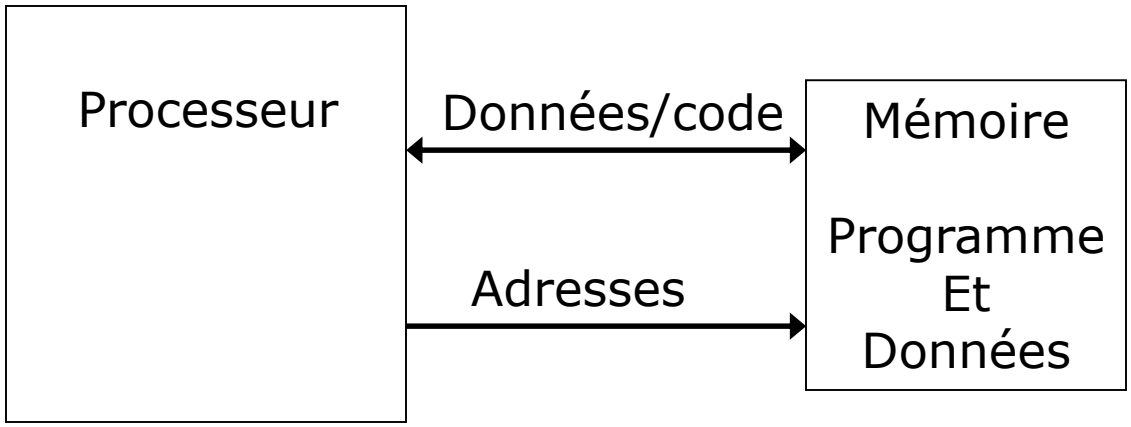
herve.le-provost@cea.fr

Exécution en pipeline des instructions



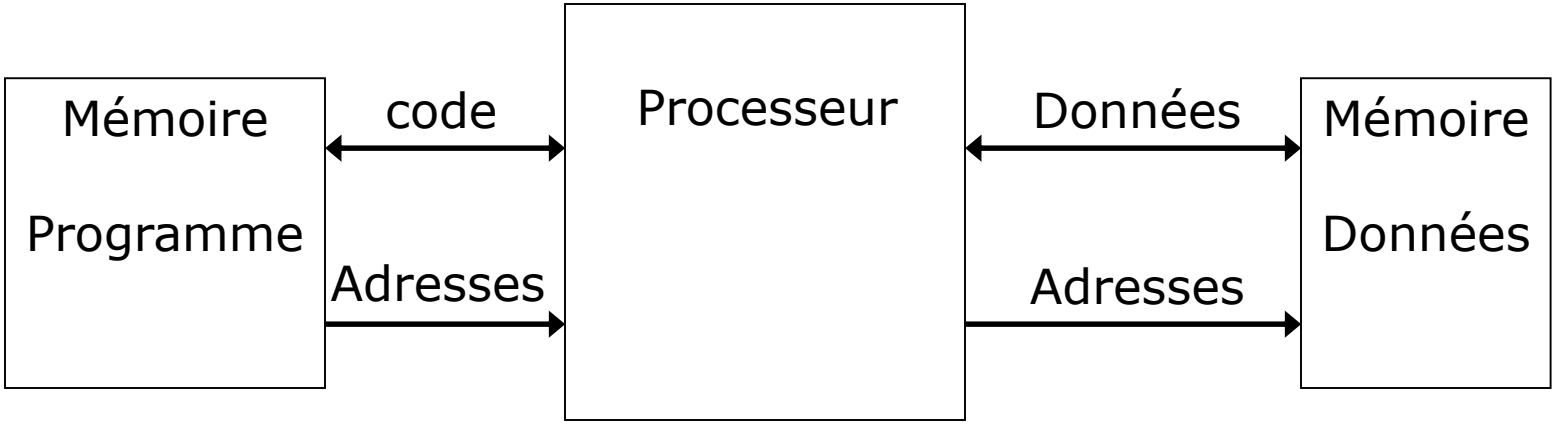
herve.le-provost@cea.fr

Processeur et Mémoires



Architecture von Neumann

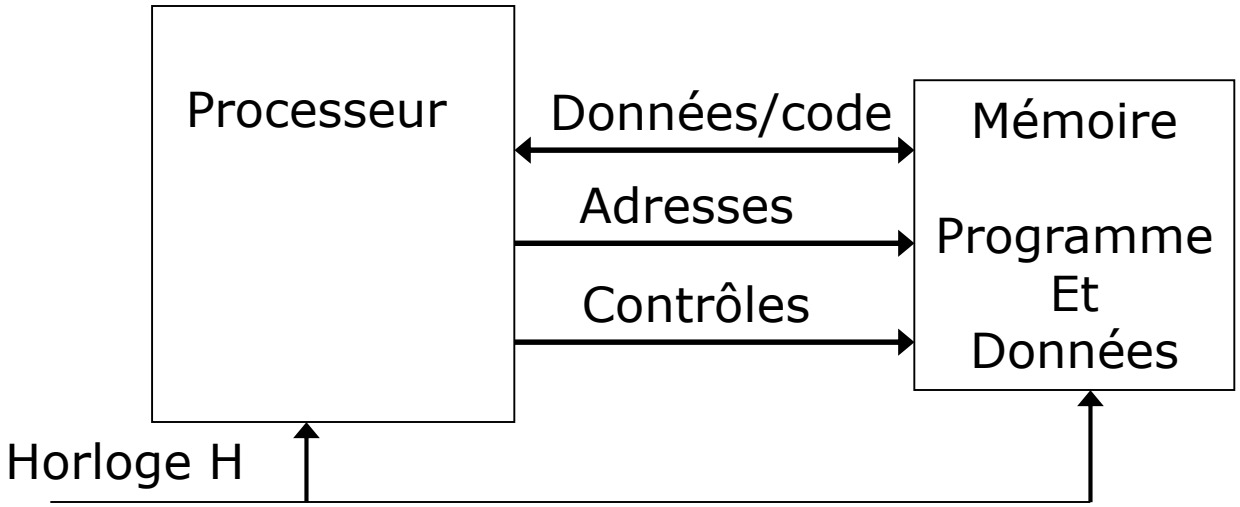
Quelle architecture est directement compatible avec une exécution en pipeline des instructions?



Architecture Harvard

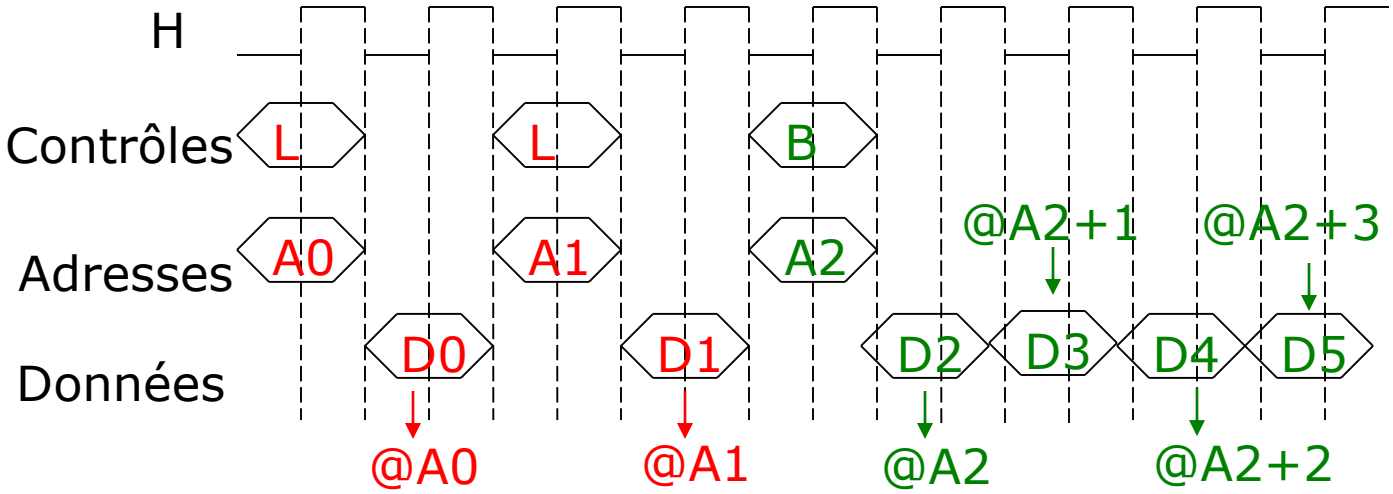
herve.le-provost@cea.fr

Bande passante mémoire



Bande Passante (BP):

Nombre maximum d'octets (données) transférable par seconde entre le processeur et la mémoire.



L : Lecture simple
F_H=100MHz
Donnée 32-bit
BP=200 Mo/s

B: Lecture multiple
FH=100MHz
Donnée 32-bit
4 données/Lecture
BP=320 Mo/s

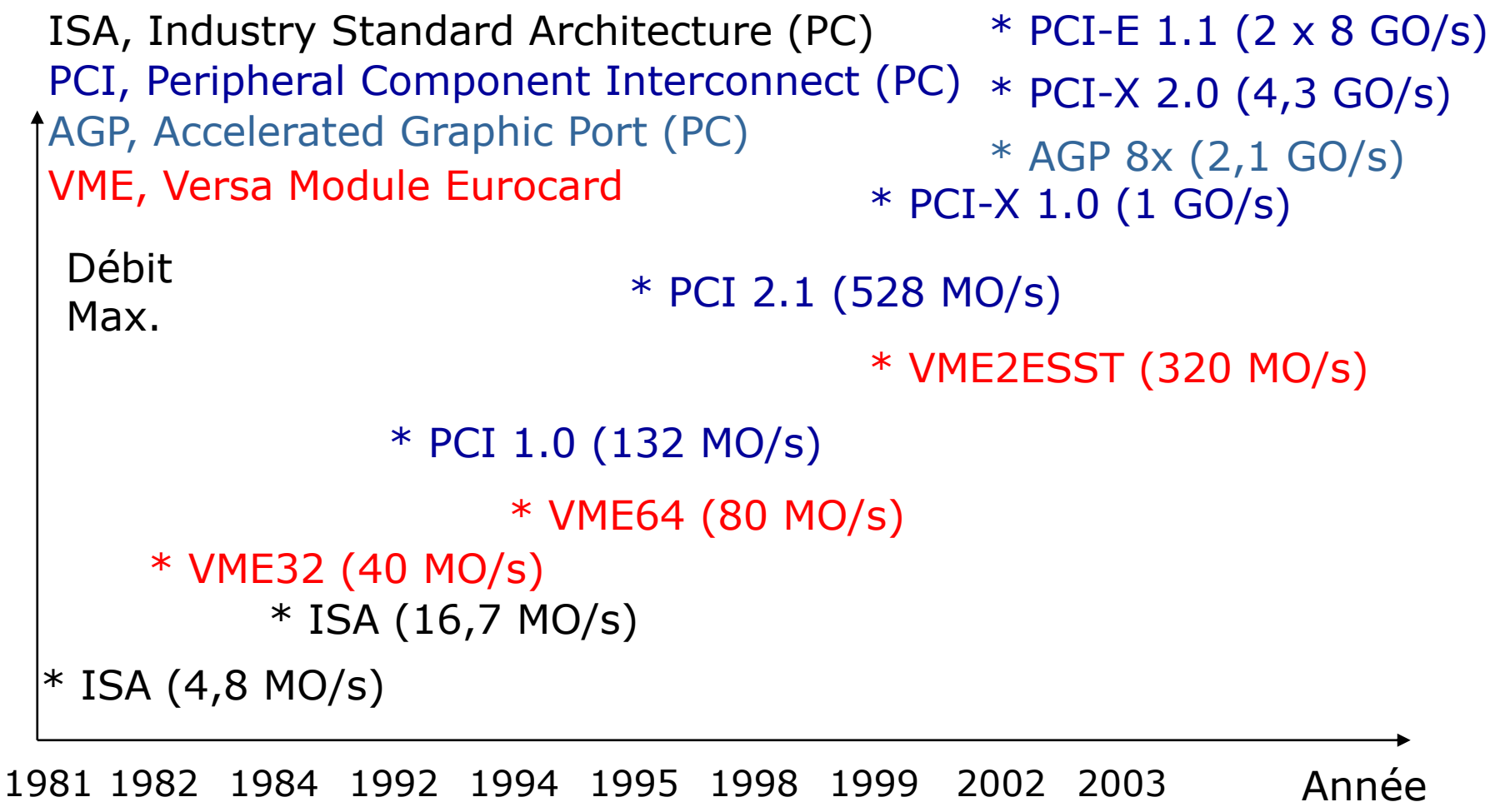
herve.le-provost@cea.fr

Calcul d'une bande passante

- Bus PCI (PC)
  - Version 1.0 32-bit/33 MHz
    - Bande passante?
  - Version 2.1 64-bit/66 MHz
    - Bande passante?
  - PCI-X 2.0 64-bit/533MHz
    - Bande passante?
- Bande passante vs Débit effectif
  - Multiplexage Adresse/Données (PCI)
  - Écriture vs lecture
  - Accès simple vs bouffée
  - Arbitrage sur le bus
  - Partage code/données (von Neumann)
  - Bande passante effective : données utiles transférées

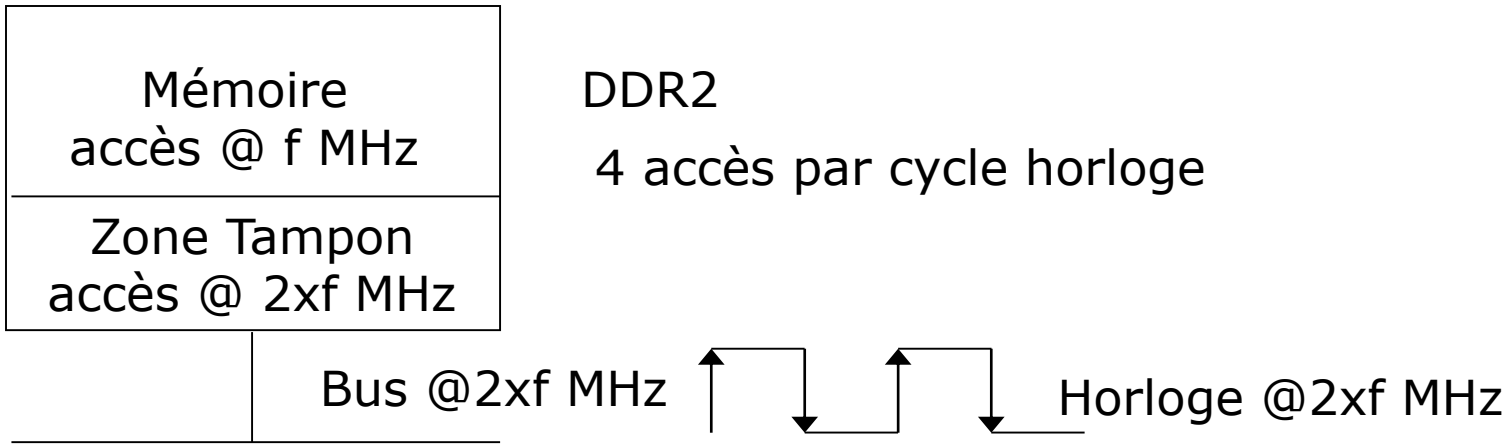
herve.le-provost@cea.fr

Évolution des Bus d'acquisition de données



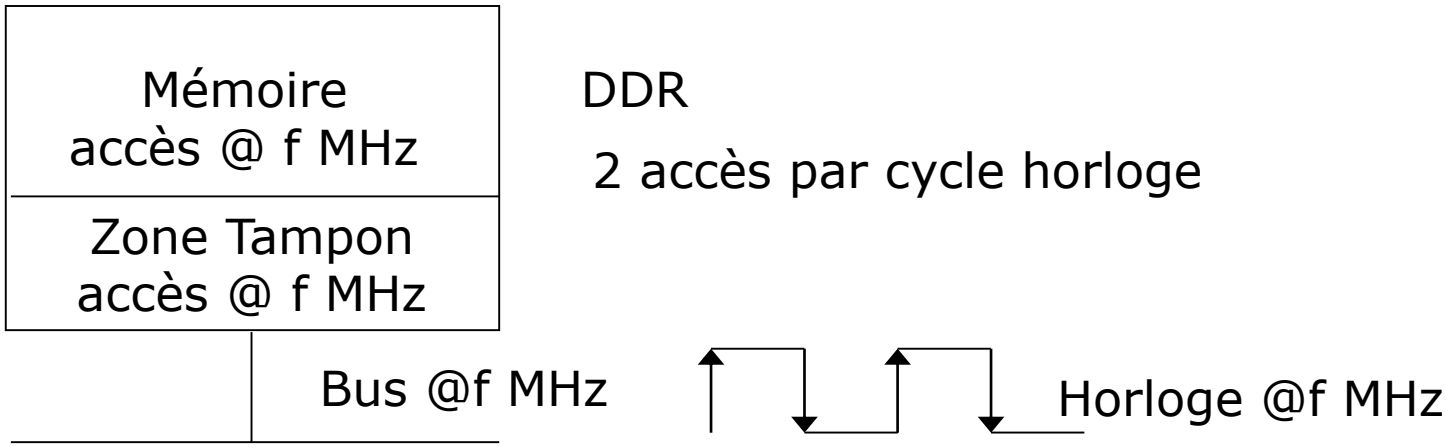
herve.le-provost@cea.fr

Mémoire SDRAM DDR2 versus SDRAM DDR



DDR2

4 accès par cycle horloge



DDR

2 accès par cycle horloge

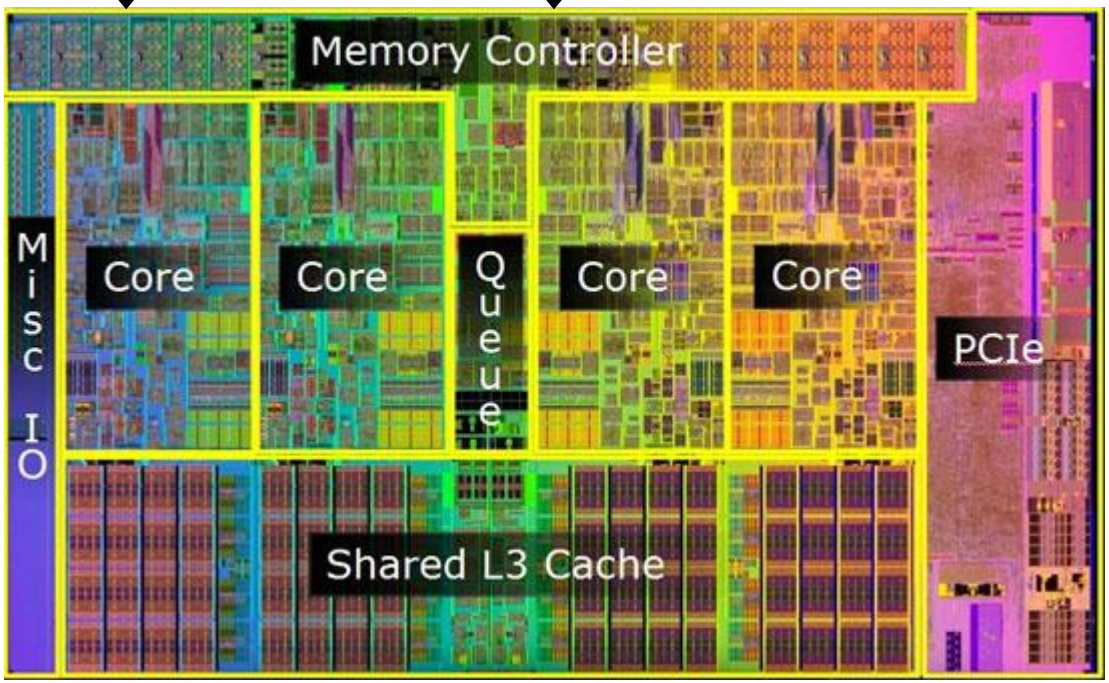
herve.le-provost@cea.fr

Exemple : le processeur Intel Core I5-700

Bande passante Mémoire 21 Go/s

DIMM DDR3 DIMM DDR3

64-bit
1333MT/s



Valeurs illustratives:
4 Cœurs @2,66GHz
Puissance 100W

Bus Adresse 36-bit (64 Go)

PCI Express

2 port x8
(liens full duplex)
ou
1 port x16
(liens full duplex)

Bande passante PCI-E
5 Gb/s par lien
(encodage 8-bit/10-bit)
Total : 16Go/s

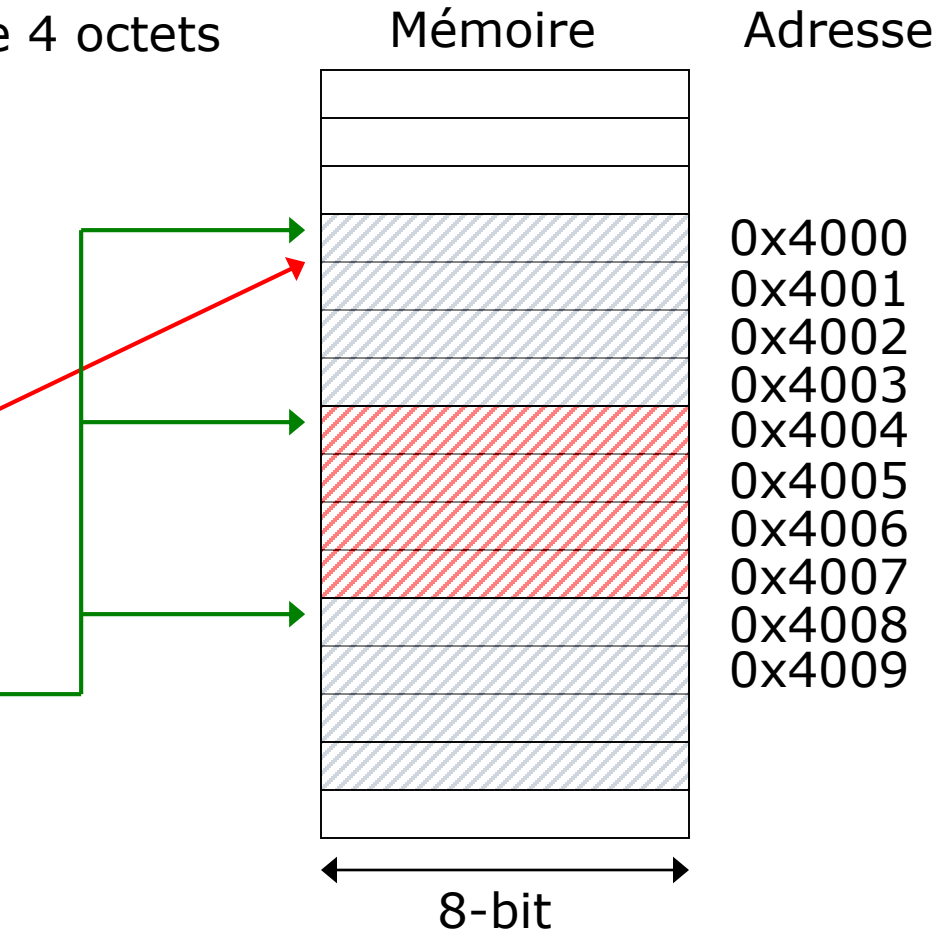
Sources: dessin www.hardware.fr Données techniques : Intel Core i7-800 and i5-700 Desktop processors series, datasheet-Volume 1, September 2009, Intel

herve.le-provost@cea.fr

Organisation mémoire

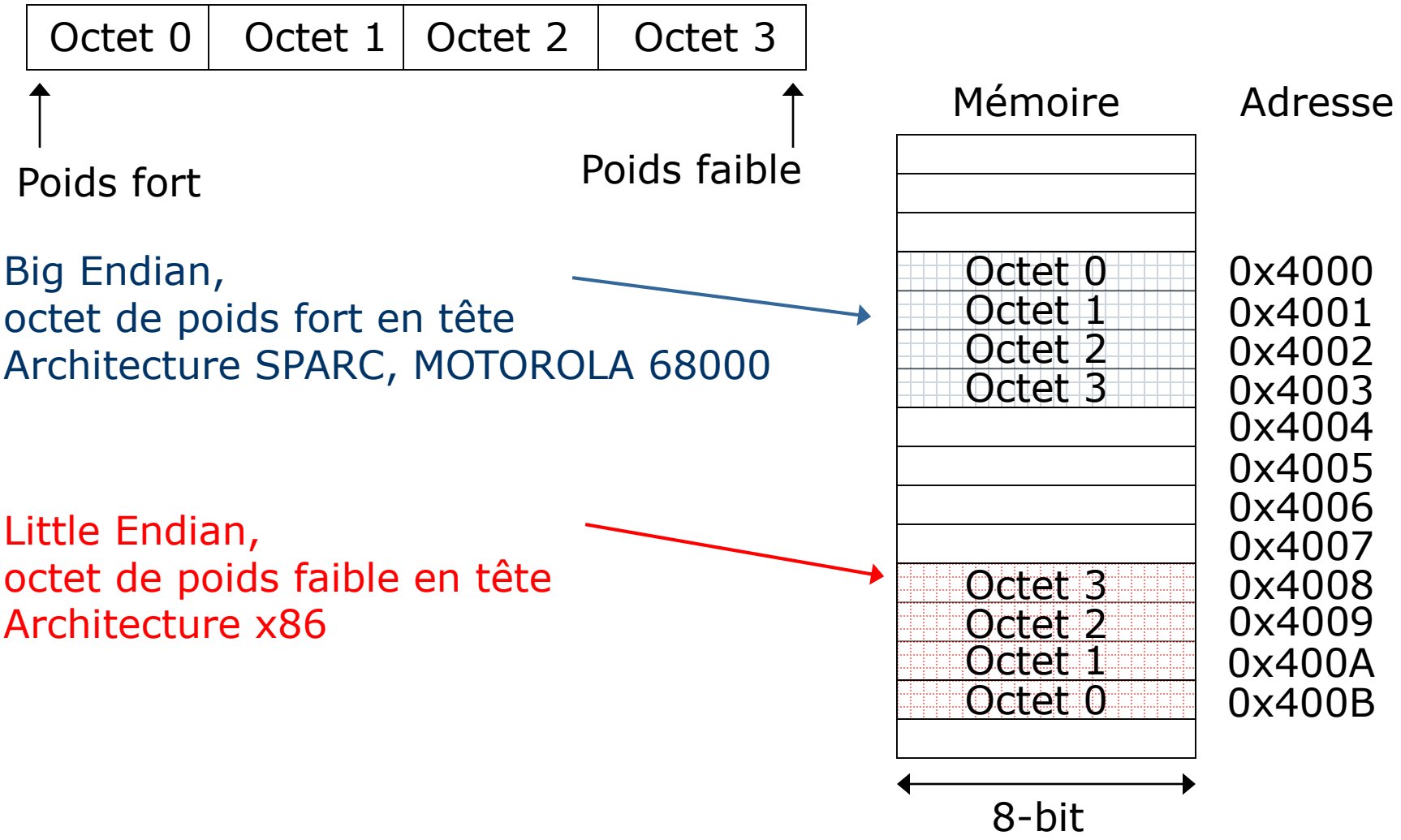
Hypothèses:
Le processeur adresse des octets
Un entier (integer) a pour taille 4 octets

```
void fonctionC()
{
int *ptr;

/*réservation 12 bytes*/
ptr=malloc(3*sizeof(int));

/* initialisation tampon */
for (i=0;i<3;i++)
    *ptr++=0;
…
}
```



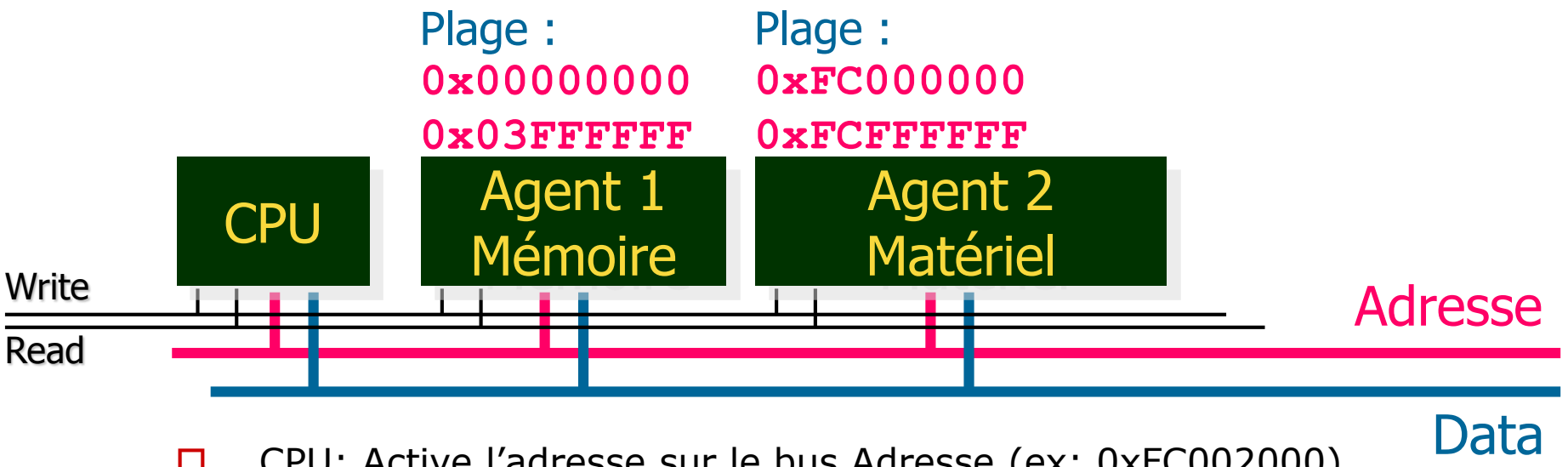
herve.le-provost@cea.fr

Organisation mémoire: Big Endian vs Little Endian



herve.le-provost@cea.fr

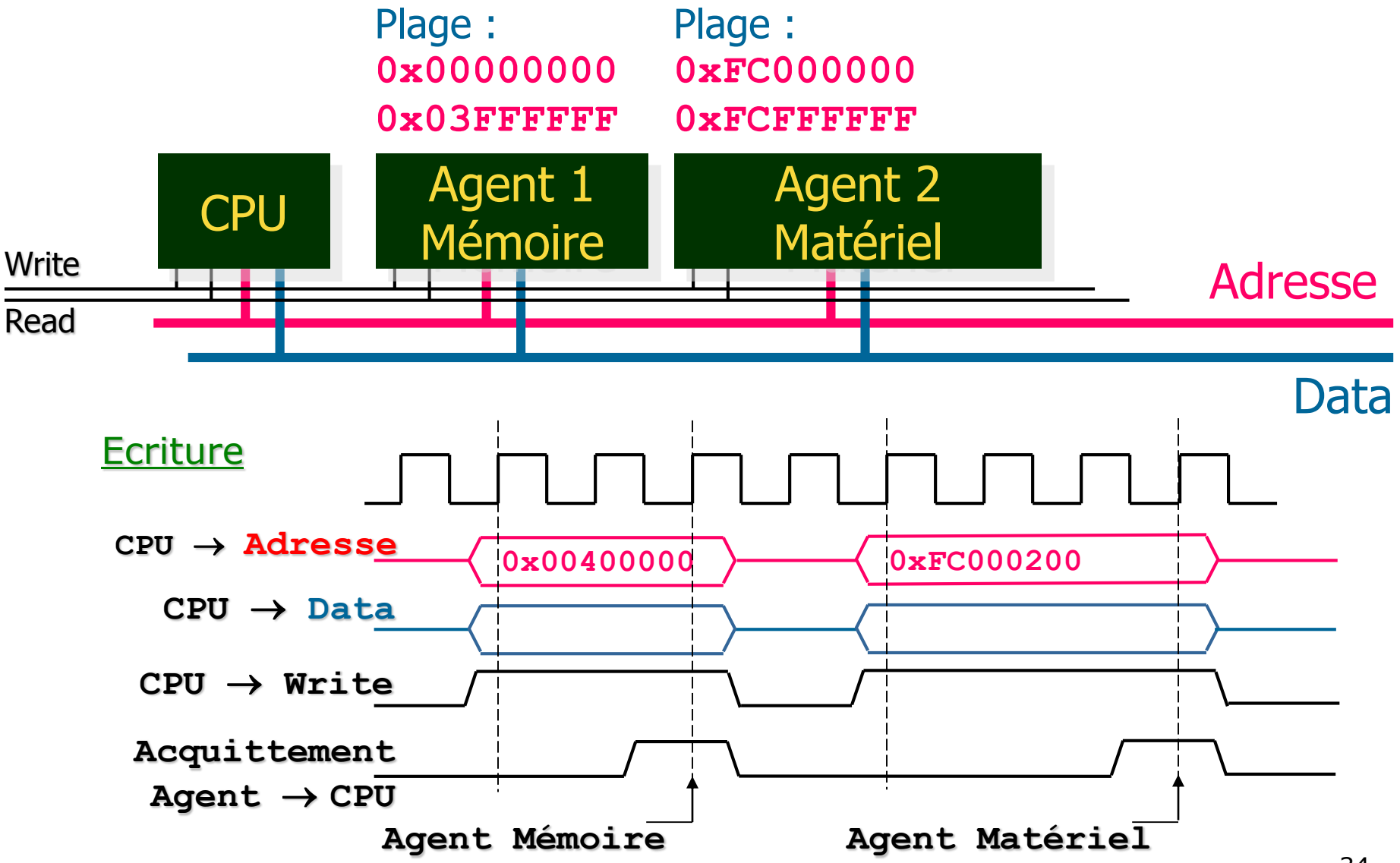
Principe de communication par bus



- CPU: Active l'adresse sur le bus Adresse (ex: 0xFC002000).
- Périphérique : un seul est sensible à cette adresse
- Si écriture :
  - CPU active la donnée sur le bus Data.
  - Périphérique : concerné traite la donnée sur le bus Data.
- Si lecture :
  - Périphérique : active la donnée sur le bus Data.
  - CPU : traite (lis) la donnée sur le bus Data.
- Modes de lecture/écriture spéciaux (rafale synchrone, etc.)
- Contrôleur mémoire : séparé ou intégré au CPU

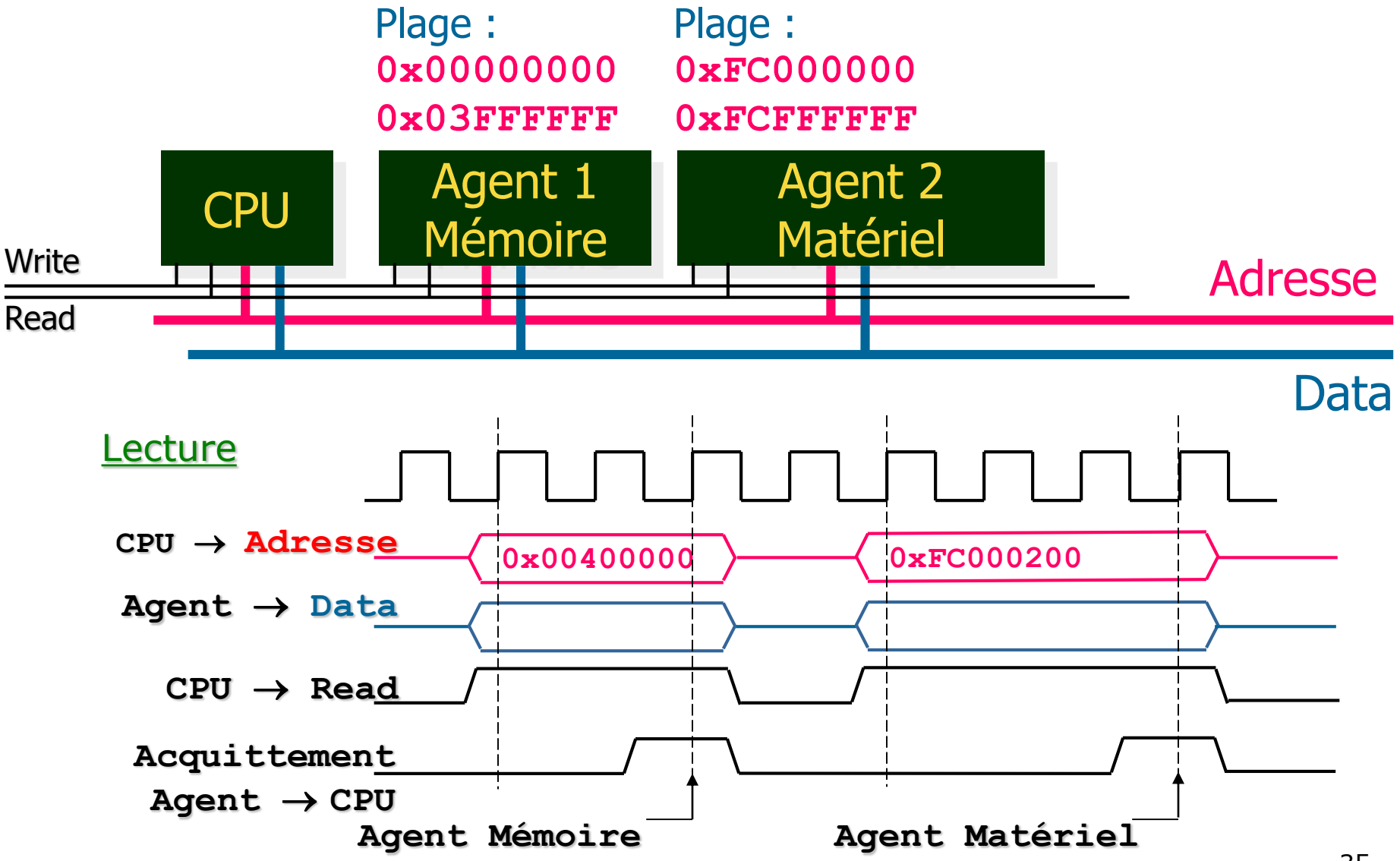
herve.le-provost@cea.fr

Principe de communication par bus



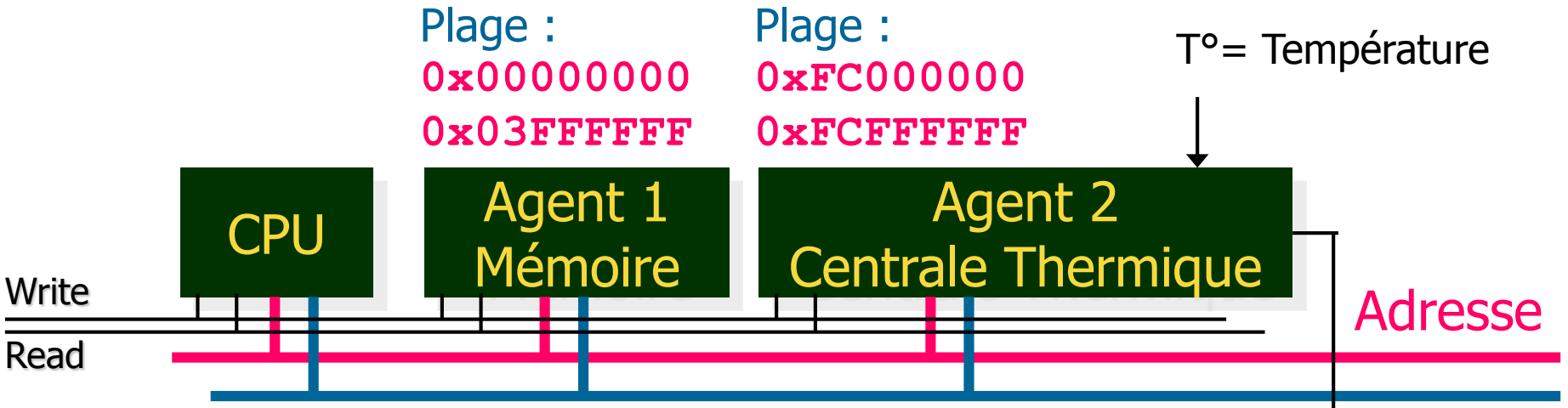
herve.le-provost@cea.fr

Principe de communication par bus



herve.le-provost@cea.fr

Principe de communication par bus



Adresse	Registre 32-bit	#bit	Mode Accès
0xFC000200	Alarme	0	Lecture/RAZ (Ecriture 1)
0xFC000204	T° courante(°C)	7 à 0	Lecture
0xFC000208	T° maximale (°C)	7 à 0	Lecture

Ecrire le code C qui sur alarme affiche
la température courante et maximale et efface l'alarme

herve.le-provost@cea.fr

Principe de communication par bus

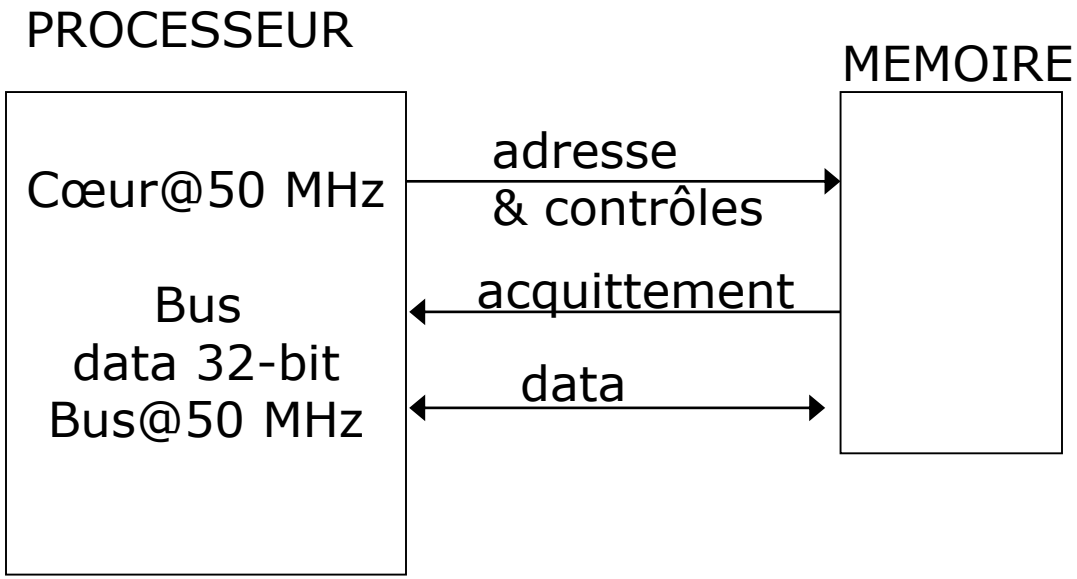
```
void alarm() {
volatile unsigned int *ptr1, *ptr2, *ptr3;
int alarme, T0, T1;
ptr1=(unsigned int*)0xFC000200;//alarme
ptr2=(unsigned int*)0xFC000204;//T° courante
ptr3=(unsigned int*)0xFC000208;//T° maximale

while (1) {
   alarme=*ptr1&1;
   if (alarme!=0) {
      T0=(*ptr2)&255;//lecture T° courante
      T1=(*ptr3)&255;//lecture T° maximale
      printf («Temp. courante %d maximale %d\n», T0, T1);
      *ptr1=1;//RAZ alarme
    }
}
}
```

Quel est l'impact de ce code sur les performances du CPU?

herve.le-provost@cea.fr

Exercices de Synthèse

Exercice 1



- 1 cycle d'adresse puis attente acquittement
- Mode rafale non supporté
- Acquittement sur le second cycle en écriture
- Acquittement sur le quatrième cycles en lecture

Calcul de la bande passante (Mo/s) effective en écriture

Calcul de la bande passante (Mo/s) effective en lecture

herve.le-provost@cea.fr

Exercices de Synthèse

Correction Exercice 1

Calcul de la bande passante effective en écriture

3 cycles@50MHz (60ns) pour transférer 4 octets
4/(60ns) = 66,7 Méga octets transférées en 1s

$BP_{Écriture}$ = 66,7 Mo/s

Calcul de la bande passante effective en lecture

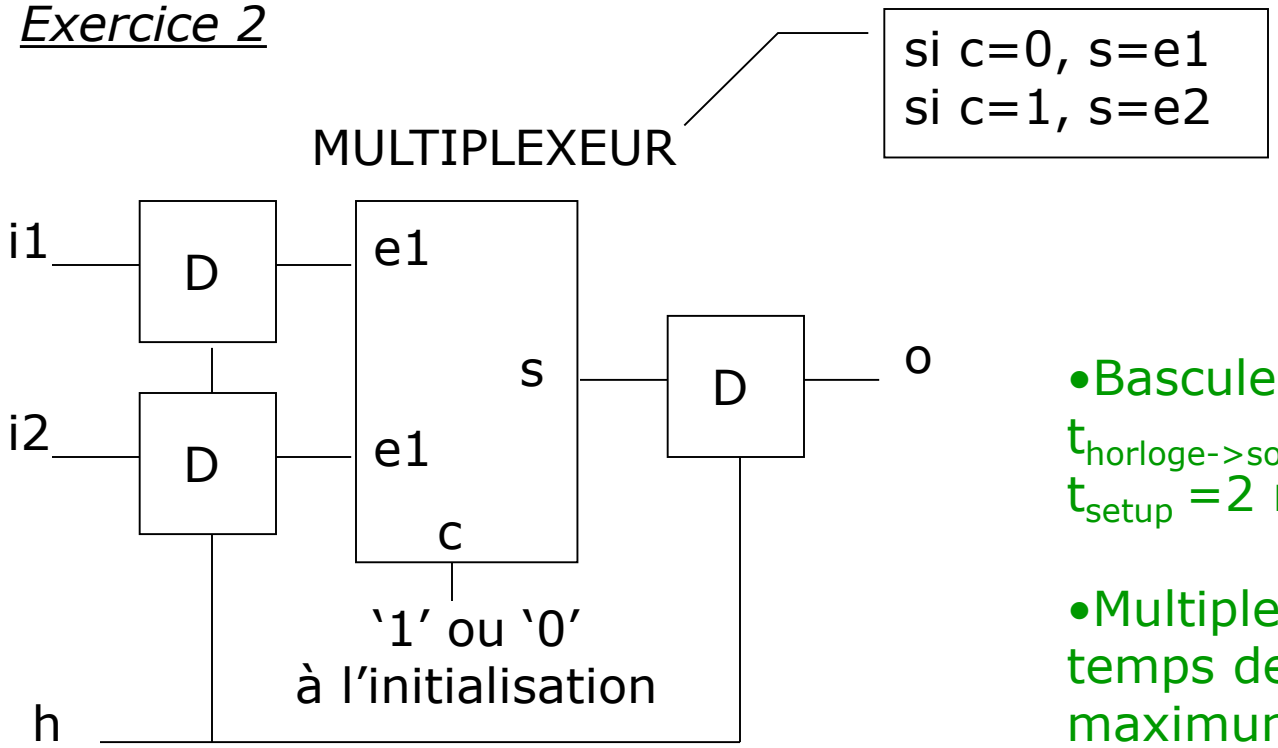
5 cycles@50MHz (100ns) pour transférer 4 octets
4/(100ns) = 40 Méga octets transférées en 1s

$BP_{Lecture}$ = 40 Mo/s

herve.le-provost@cea.fr

Exercices de Synthèse

Exercice 2



- Bascule D:
  $t_{horloge->sortie}$ =3 ns maximum
  t_{setup} =2 ns minimum

- Multiplexeur:
  temps de propagation 23 ns maximum

Calcul de la fréquence maximale de fonctionnement

herve.le-provost@cea.fr

Exercices de Synthèse

Correction Exercice 2

Calcul de la fréquence maximale de fonctionnement

$F_{max} = 1/(t_{horloge->sortie} + t_{setup} + t_{prop.\ Mux.})$

$F_{max} = 1/(3\ ns + 2\ ns + 23\ ns)$

$F_{max} = 35{,}7\ MHz$

herve.le-provost@cea.fr

Les interruptions

- L'autre moyen de communication entre un périphérique et le CPU est l'interruption matérielle. C'est la seule communication qui soit ***initiée à l'initiative du périphérique (esclave)***.

- Son fonctionnement exact dépend de l'architecture matérielle de la carte et des liaisons entre le périphérique et les broches d'interruption du CPU.

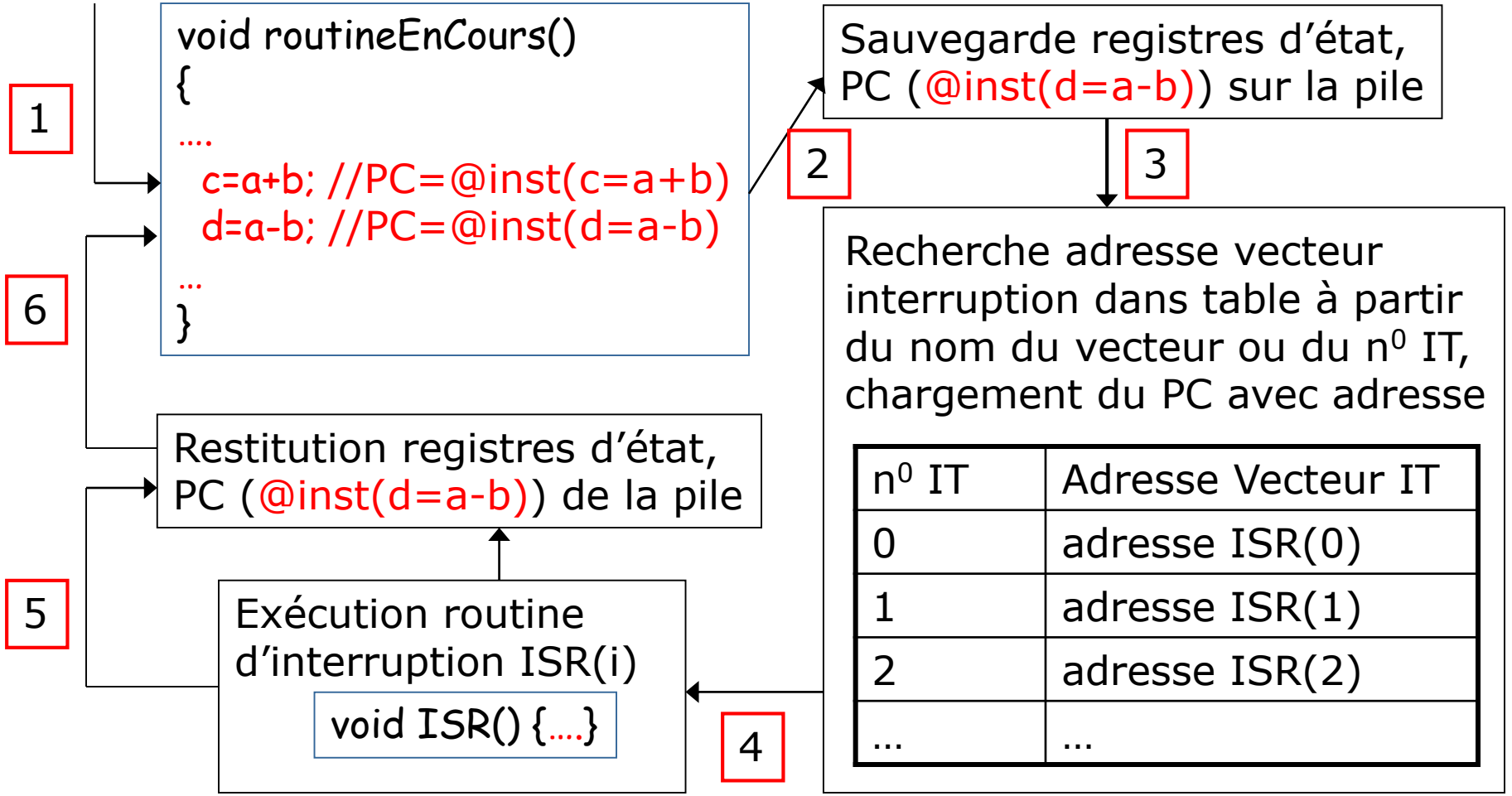
- Le CPU associe à une broche une fonction d'interruption. Dans cette fonction, l'interruption est traitée. Si le CPU est animé par un OS :
  - Tous les appels susceptibles d'être bloquants sont interdits au sein de la fonction d'interruption.
  - Le temps d'exécution de l'interruption doit être minimisé. Tout traitement lourd doit être effectué par une tâche standard en attente de libération d'un sémaphore. Ce sémaphore est libéré par la fonction d'interruption

herve.le-provost@cea.fr

Les interruptions

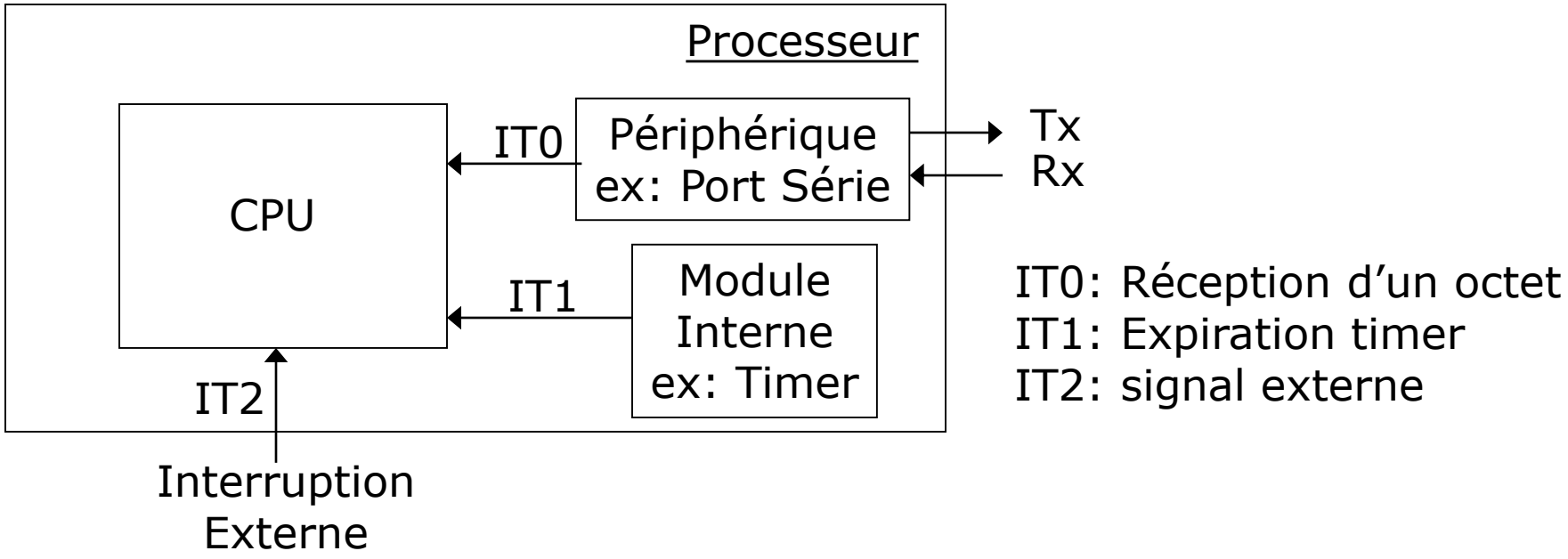
Interruption (IT) matérielle/logicielle n^0 i

ISR : Interrupt Service Routine
PC : Program Counter



herve.le-provost@cea.fr

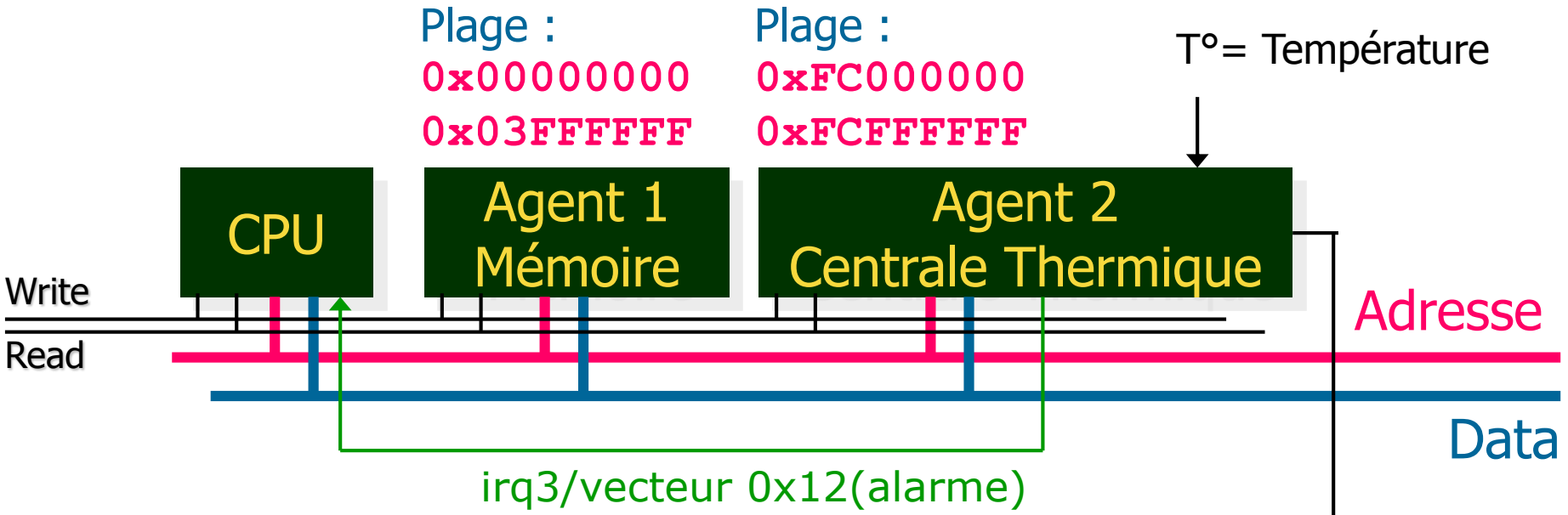
Les interruptions



- Interruption externe asynchrone, sélection front montant, état
- Trois principaux registres d'interruption:
  - Registre d'autorisation individuelle
  - Registre de présence individuelle
  - Registre de remise à zéro individuelle
- Priorités entre les interruptions définies par le processeur cible
- Sur interruption, sauvegarde automatique du PC et des registres d'état sur la pile

herve.le-provost@cea.fr

Les interruptions



Adresse	Registre 32-bit	#bit	Mode Accès
0xFC000204	T° courante(°C)	7 à 0	Lecture
0xFC000208	T° maximale (°C)	7 à 0	Lecture

herve.le-provost@cea.fr

Les interruptions

```
//Exemple gestion interruptions-Système exploitation RTEMS
void initSystem() {
rtems_status_code status;
rtems_isr_entry old_handle;
//…
status=rtems_interrupt_catch(alarmCatch, 0x12,&old_handle);
//…
}
rtems_isr alarmCatch(rtems_vector_number vector) {
volatile unsigned int  *ptr2, ptr3;
int T0, T1;

ptr2=(unsigned int*)0xFC000204;//T° courante
ptr3=(unsigned int*)0xFC000208;//T° maximale
T0=(*ptr2)&255;//lecture T° courante
T1=(*ptr3)&255;//lecture T° maximale
 printk («Temp. courante %d maximale %d\n», T0, T1);
}
```

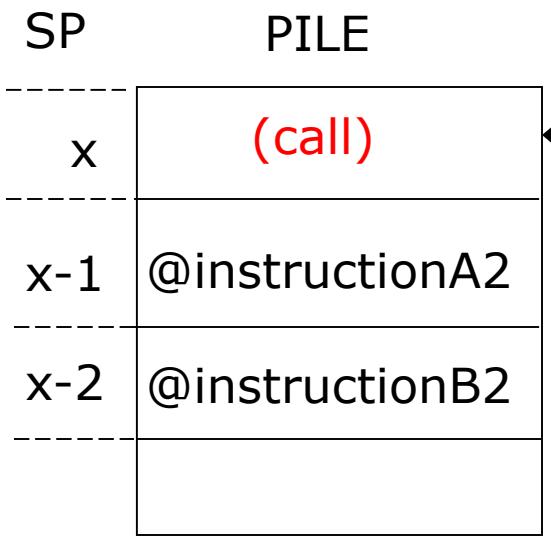

herve.le-provost@cea.fr

La Pile: Principe

```
routine_A {
instruction A1;
…
call routine_B ;
instruction A2;
….}
```

```
routine_B {
instruction B1;
…
call routine_C ;
instruction B2;
….
return;}
```

```
routine_C {
instruction C1;
…
return;}
```



SP Haut de la pile

call routine_B : SP<=SP-1
1/empiler PC 2/PC<=@instructionB1

call routine_C : SP<=SP-1
1/empiler PC 2/PC<=@instructionC1

SP Haut de la pile

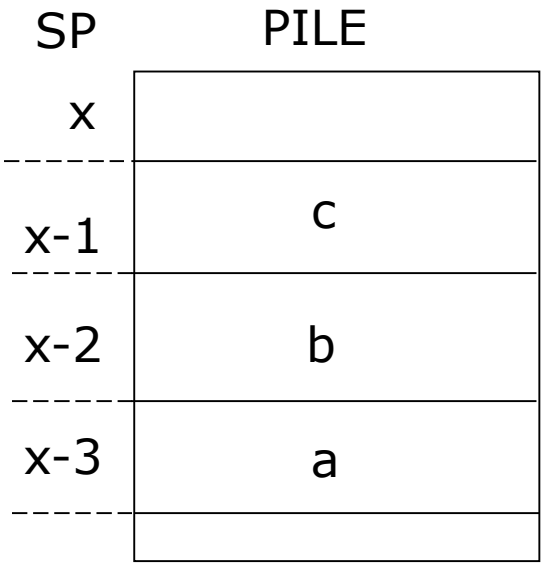
return routine_B
1/PC<=@instructionA2 2/SP<=SP+1

return routine_C
1/PC<=@instructionB2 2/SP<=SP+1

herve.le-provost@cea.fr

La Pile : variables automatiques

```
int main()
{
int a=3,b=5,c;

c=a+b;

return 0;
}
```



SP Haut de la pile

1. Décrémentation SP de 3/Réservation mémoire pour ‘a’,’b’,’c’
2. Stockage valeurs ‘a’,’b’ sur la pile
3. Accès aux variables depuis la pile pour exécution du calcul
4. Stockage résultat ‘c’ sur la pile
5. Incrémentation SP de 3/Libération mémoire

herve.le-provost@cea.fr

La Pile : variables automatiques

```
;codage assembleur DSP TMS320C55
;int main()
;{
01418F                  main:
01418F 4efd             AADD #-3,SP
;int a=3,b=5,c;
014191 e60003           MOV #3,*SP(#00h)
014194 e60205           MOV #5,*SP(#01h)
;c=a+b;
014197 a902             MOV *SP(#01h),AR1
014199 d60099           ADD *SP(#00h),AR1,AR1
01419C c904             MOV AR1,*SP(#02h)
;return 0;
01419E 3c04             MOV #0,T0
;}
0141A0 4e03             AADD #3,SP
0141A2 4804             RET
```

Adresse instruction

Code machine

Décrémentation pointeur de pile

Stockage valeurs a et b sur la pile

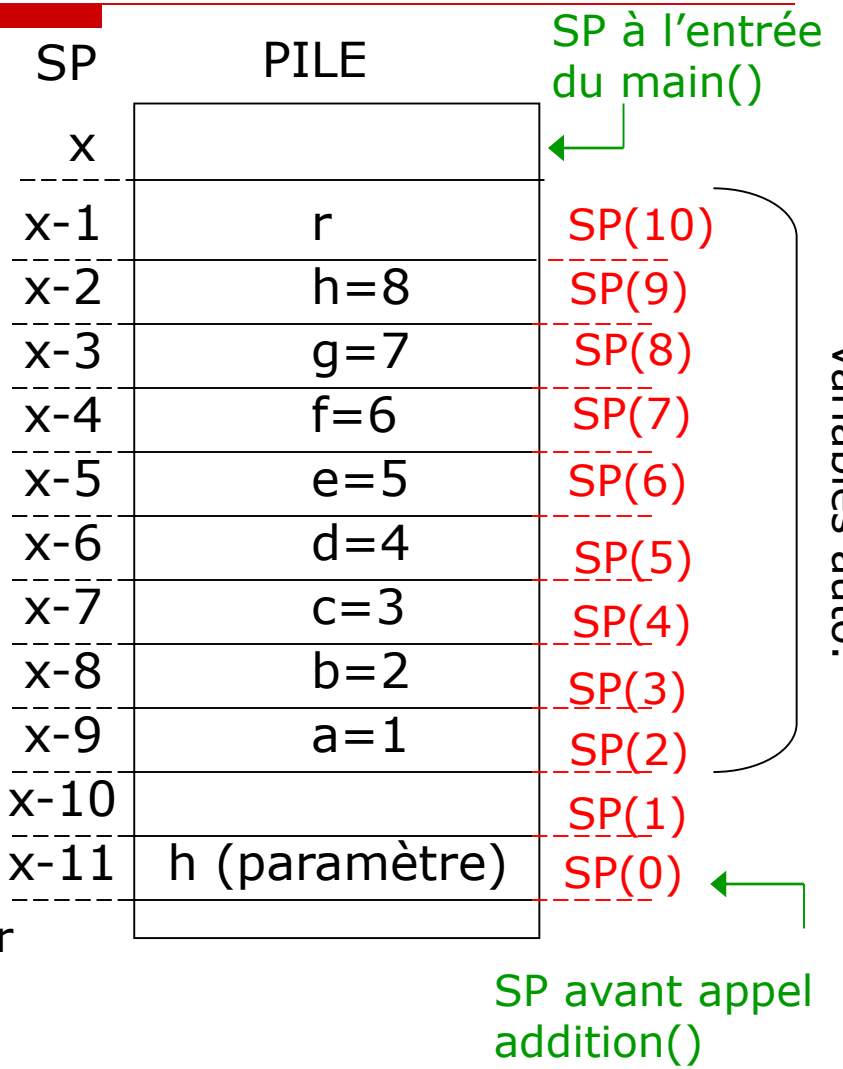
Calcul

Stockage résultat Sur la pile

herve.le-provost@cea.fr

La Pile : passage des paramètres

```
int main()
{
int a=1,b=2,c=3,d=4,e=5,f=6,g=7,h=8;
int r;
        r=addition(a,b,c,d,e,f,g,h);
        return 0;
}
int addition(int a,int b,int c,int d,int e,
int f,int g,int h)
{
int r;
        r=a+b+c+d+e+f+g+h;
        return r;
}
```



- Paramètres a, b, c, d, e, f, g passés par registre, paramètre h passé sur la pile
- Retour r passé par registre
- Dépendant du couple compilateur/cible, du niveau d'optimisation du compilateur

herve.le-provost@cea.fr

La Pile : passage des paramètres

```
01418F 4ef5          AADD #-11,SP
014191 e60401        MOV #1,*SP(#02h)
014194 e60602        MOV #2,*SP(#03h)
014197 e60803        MOV #3,*SP(#04h)
01419A e60a04        MOV #4,*SP(#05h)
01419D e60c05        MOV #5,*SP(#06h)
0141A0 e60e06        MOV #6,*SP(#07h)
0141A3 e61007        MOV #7,*SP(#08h)
0141A6 e61208        MOV #8,*SP(#09h)
0141A9 a912          MOV *SP(#09h),AR1
0141AB c900          MOV AR1,*SP(#00h)
0141AD a404          MOV *SP(#02h),T0
0141AF a506          MOV *SP(#03h),T1
0141B1 a808          MOV *SP(#04h),AR0
0141B3 aa0c          MOV *SP(#06h),AR2
0141B5 ab0e          MOV *SP(#07h),AR3
0141B7 ac10          MOV *SP(#08h),AR4
0141B9 a90a          MOV *SP(#05h),AR1
0141BB 080008        CALL addition
0141BE c414          MOV T0,*SP(#0ah)
0141C0 3c04          MOV #0,T0
0141C2 4e0b          AADD #11,SP
0141C4 4804          RET
```

Décrémentation pointeur de pile

Stockage valeurs de 8 variables sur la pile

Passage de 1 paramètre par la pile

Passage de 7 paramètres par registre

Appel addition

Stockage résultat sur la pile

herve.le-provost@cea.fr

La Pile : passage des paramètres

```
0141C6            addition:
0141C6 4ef7               AADD #-9,SP
0141C8 cc0c               MOV AR4,*SP(#06h)
0141CA cb0a               MOV AR3,*SP(#05h)
0141CC ca08               MOV AR2,*SP(#04h)
0141CE c906               MOV AR1,*SP(#03h)
0141D0 c804               MOV AR0,*SP(#02h)
0141D2 c502               MOV T1,*SP(#01h)
0141D4 c400               MOV T0,*SP(#00h)
0141D6 a902               MOV *SP(#01h),AR1
0141D8 d60099             ADD *SP(#00h),AR1,AR1
0141DB d60499             ADD *SP(#02h),AR1,AR1
0141DE d60699             ADD *SP(#03h),AR1,AR1
0141E1 d60899             ADD *SP(#04h),AR1,AR1
0141E4 d60a99             ADD *SP(#05h),AR1,AR1
0141E7 d60c99             ADD *SP(#06h),AR1,AR1
0141EA d61499             ADD *SP(#0ah),AR1,AR1
0141ED c90e               MOV AR1,*SP(#07h)
0141EF 2294               MOV AR1,T0
0141F1 4e09               AADD #9,SP
0141F3 4804               RET
```

Décrémentation pointeur de pile

Transfert paramètres des registres vers la pile

Addition paramètres passés par registre

Addition paramètres passé par la pile

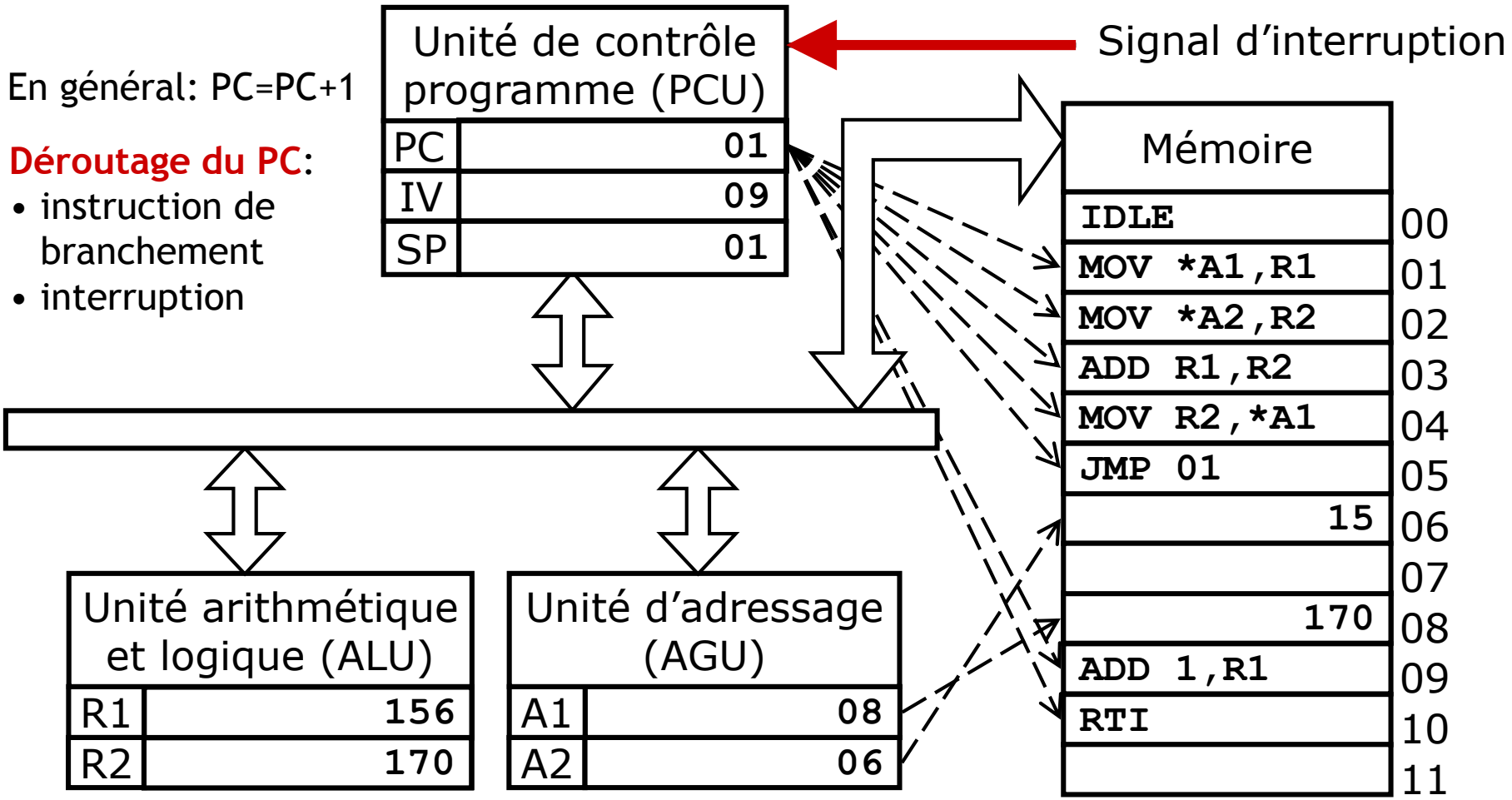
Stockage résultat sur la pile

Retour résultat par registre

herve.le-provost@cea.fr

Fonctionnement du processeur

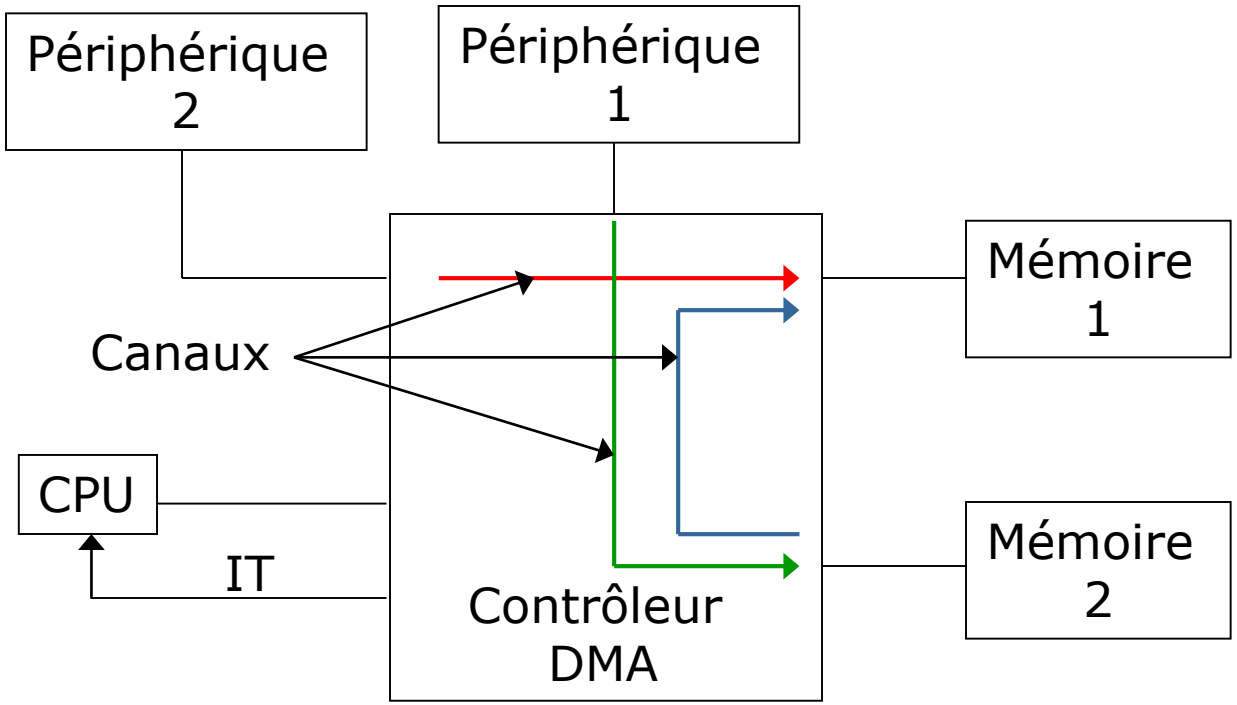
Animation PowerPoint



© Shebli Anvar

herve.le-provost@cea.fr

DMA (Direct Memory Access): principe



CPU → 1 Configuration :
- Port-adresse source
- Port-adresse destination
- Nombre d'éléments à transférer

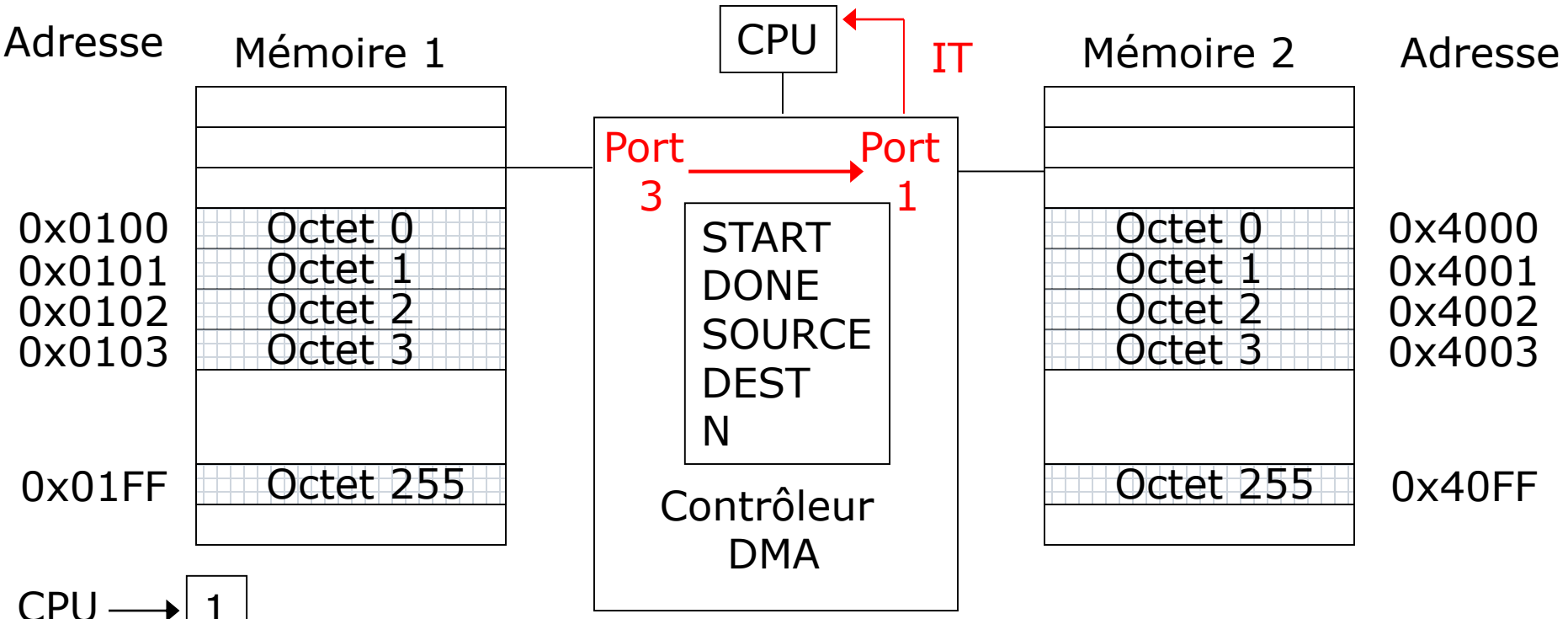
CPU → 2 Démarrage transfert

DMA → 3 Fin du transfert
Interruption CPU optionnelle

DMA → 4 Redémarrage automatique
Ou attente d'un nouveau démarrage ordonné par le CPU

herve.le-provost@cea.fr

DMA: exemple



CPU ⟶ 1

- Source Port=3 @=0x0100
- Dest. Port=1 @=0x4000
- N=256

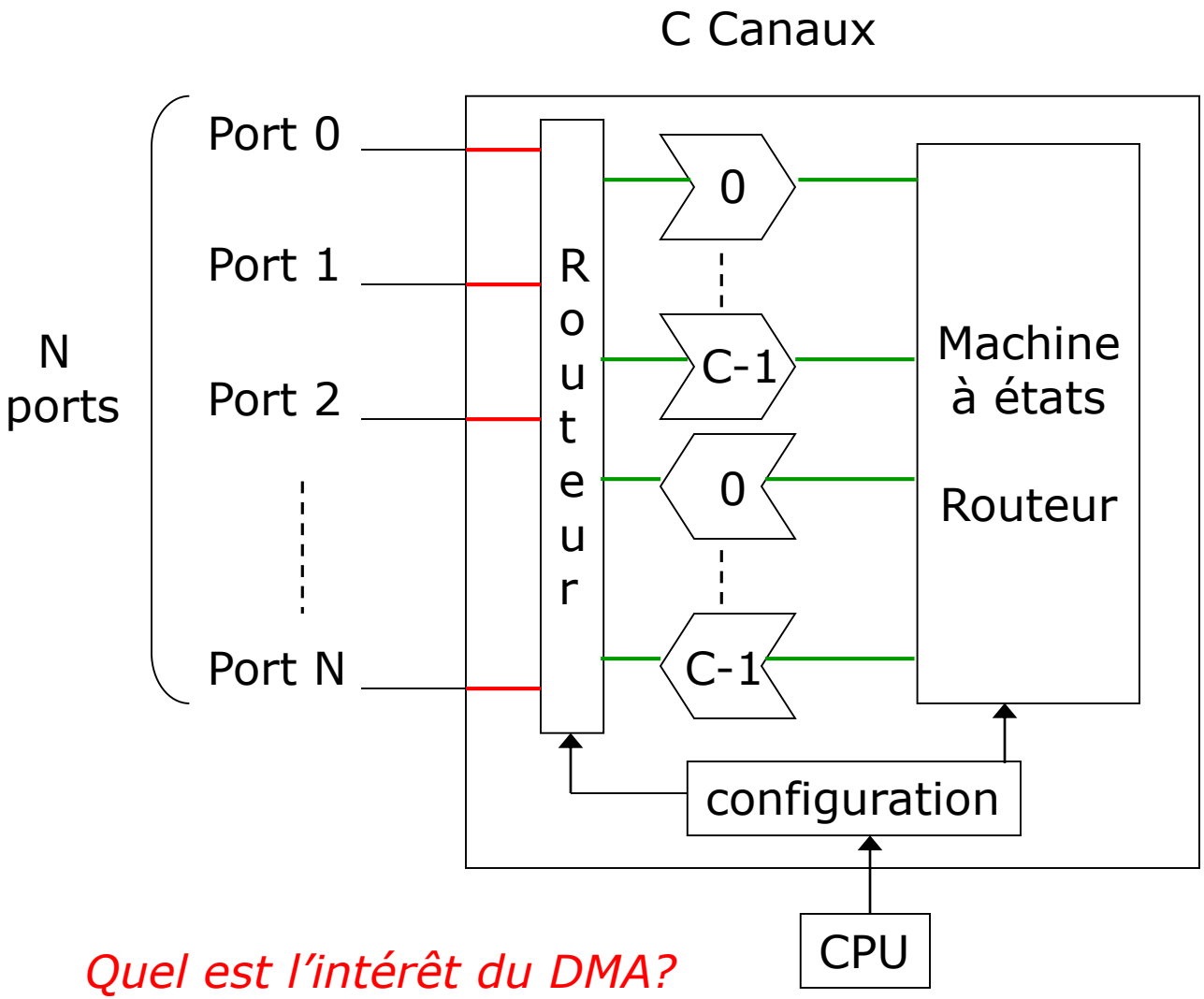
CPU ⟶ 2 START=1

DMA ⟶ 3 DONE=1; IT=1

DMA ⟶ 4 Attente Nouveau transfert

herve.le-provost@cea.fr

DMA: structure interne



- Limitation du nombre de canaux
- Distinction entre canaux et ports
- Port connectable sur lui-même

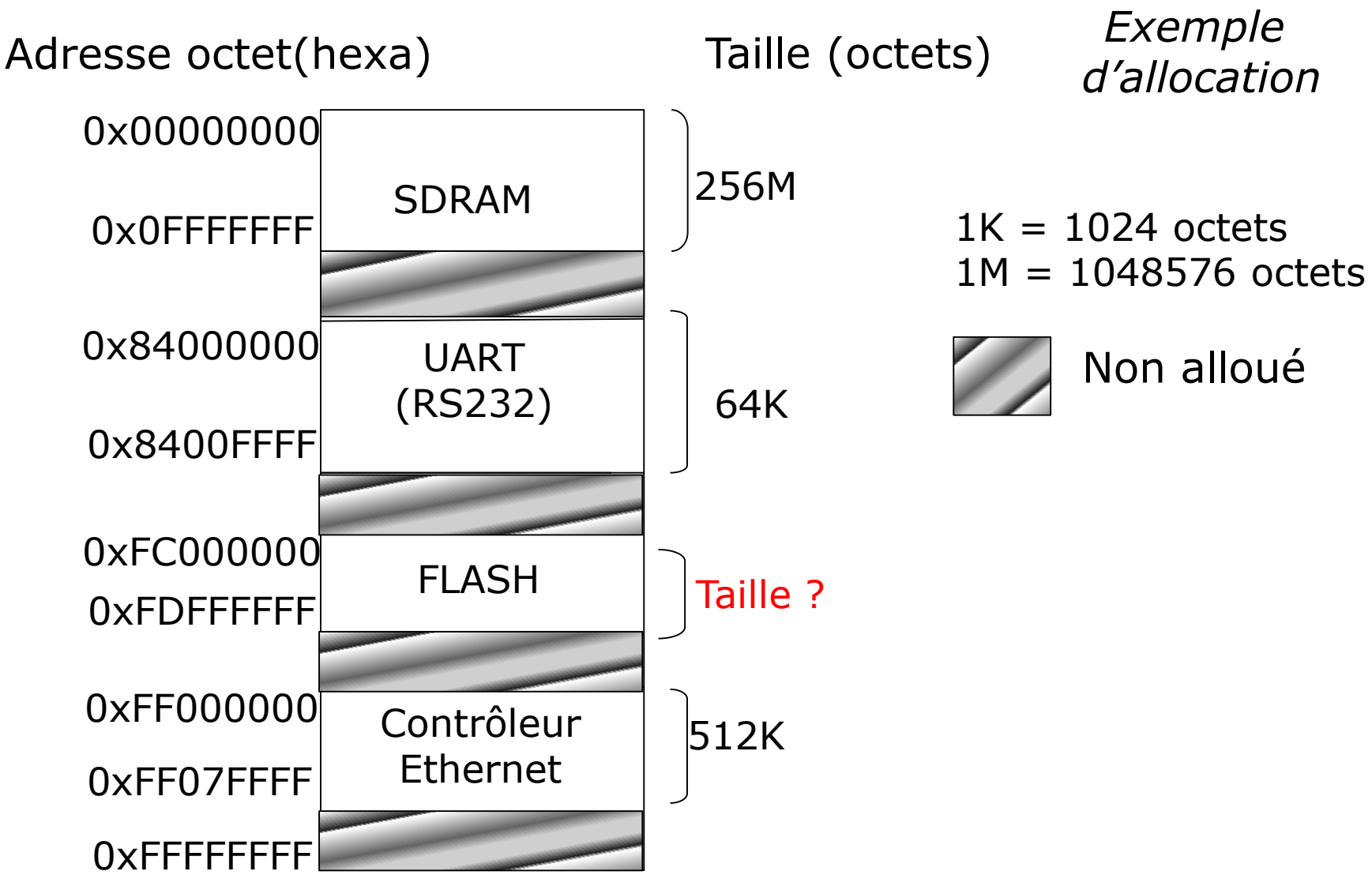
Quel est l'intérêt du DMA?

herve.le-provost@cea.fr

- Maîtrise fine du système:
  - Matériel
    - La bande passante et la gestion de la mémoire
    - La pile
    - Le pipeline des données et des instructions
    - La fréquence de fonctionnement
  - Logiciel
    - La chaine de compilation et ses niveaux d'optimisation et interactions avec le matériel.
    - Le langage assembleur pour les portions critiques
- Les éléments de performance
  - Les interruptions
  - Le DMA
  - *La mémoire cache* (seconde partie du cours)

herve.le-provost@cea.fr

La mémoire : les plages (vue du processeur)



herve.le-provost@cea.fr

Systèmes multitâches

- Appel régulier d'un **ordonnanceur**
  - Régularité assurée par une alarme électronique (hardware timer)
  - TICK = période de cette alarme (typiquement ~10 ms)
- Préemption: appel de l'ordonnanceur par **interruption**
  - L'alarme active un signal d'interruption
  - La routine associée à l'alarme d'ordonnancement est le code de l'ordonnanceur
- Le code de l'ordonnanceur « active » une tâche
  - Considère la liste des données liées à la gestion des tâches
  - Sauvegarde le contexte de la tâche en cours
  - Décide par un algorithme de la tâche à activer
  - Restaure le contexte de la tâche élue
  - Contexte $\supseteq$ valeurs des registres du processeur, notamment le PC

herve.le-provost@cea.fr