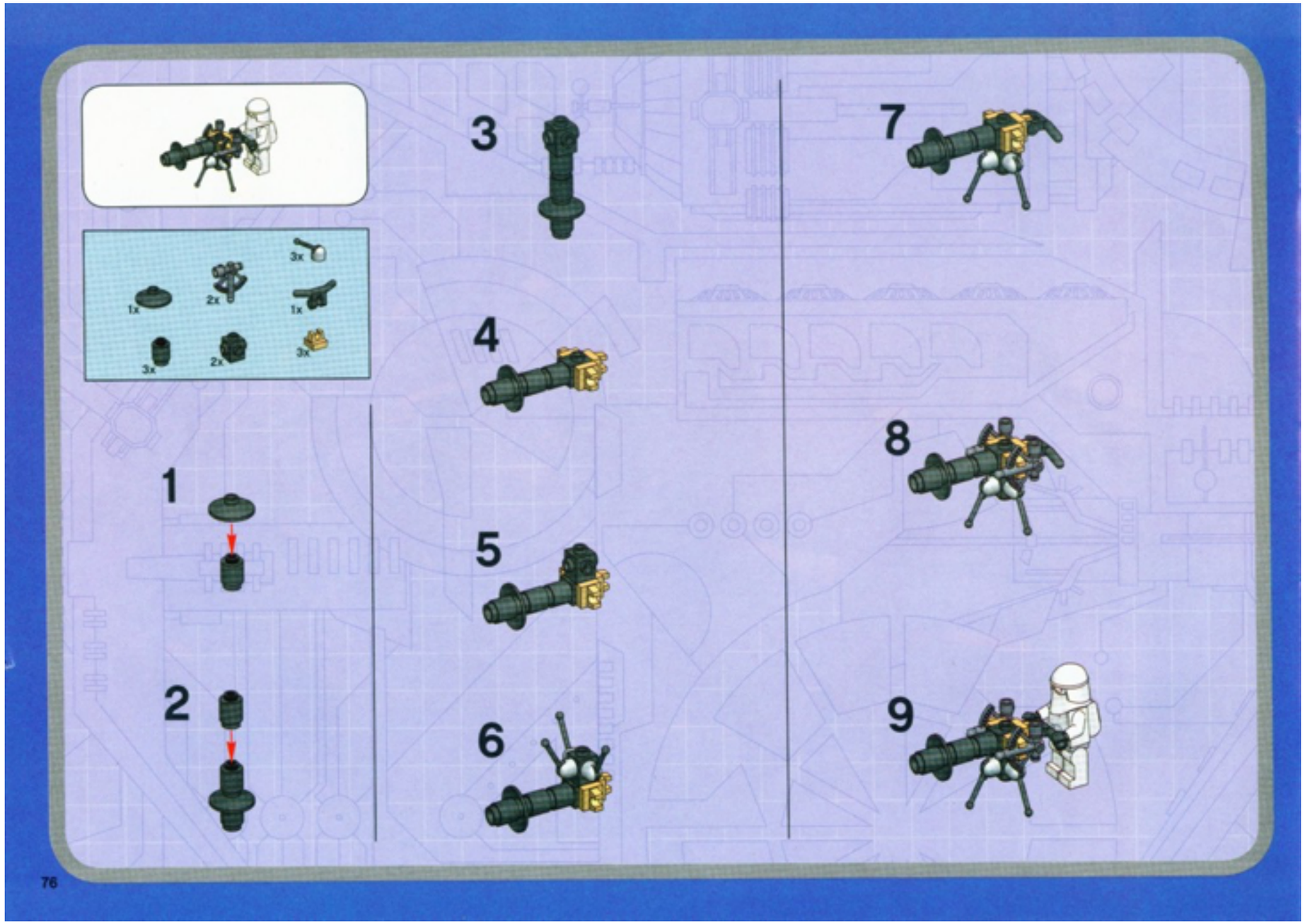
Micro-instructions



Ordinateurs: Structure et Applications, Hiver 2016

Jean-François Lalonde

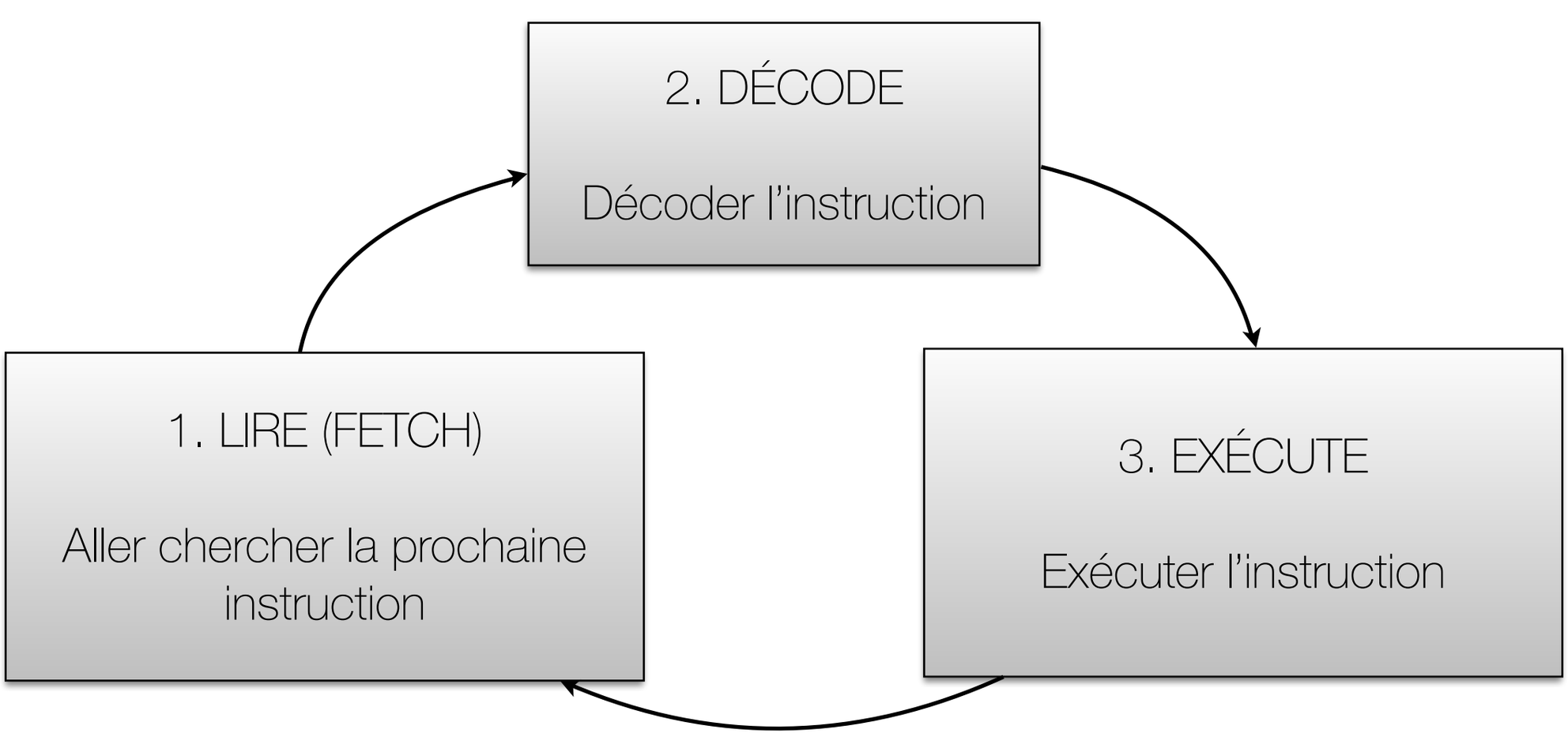
Instructions vs micro-instructions

- Une instruction est une « action » pouvant être exécutée par le microprocesseur.
- Une instruction n’est pas atomique: elle est divisée en une séquence de « micro-instructions » (µ-instructions)
- Ces µ-instructions dépendent de la structure interne, et du cycle d’instructions du microprocesseur

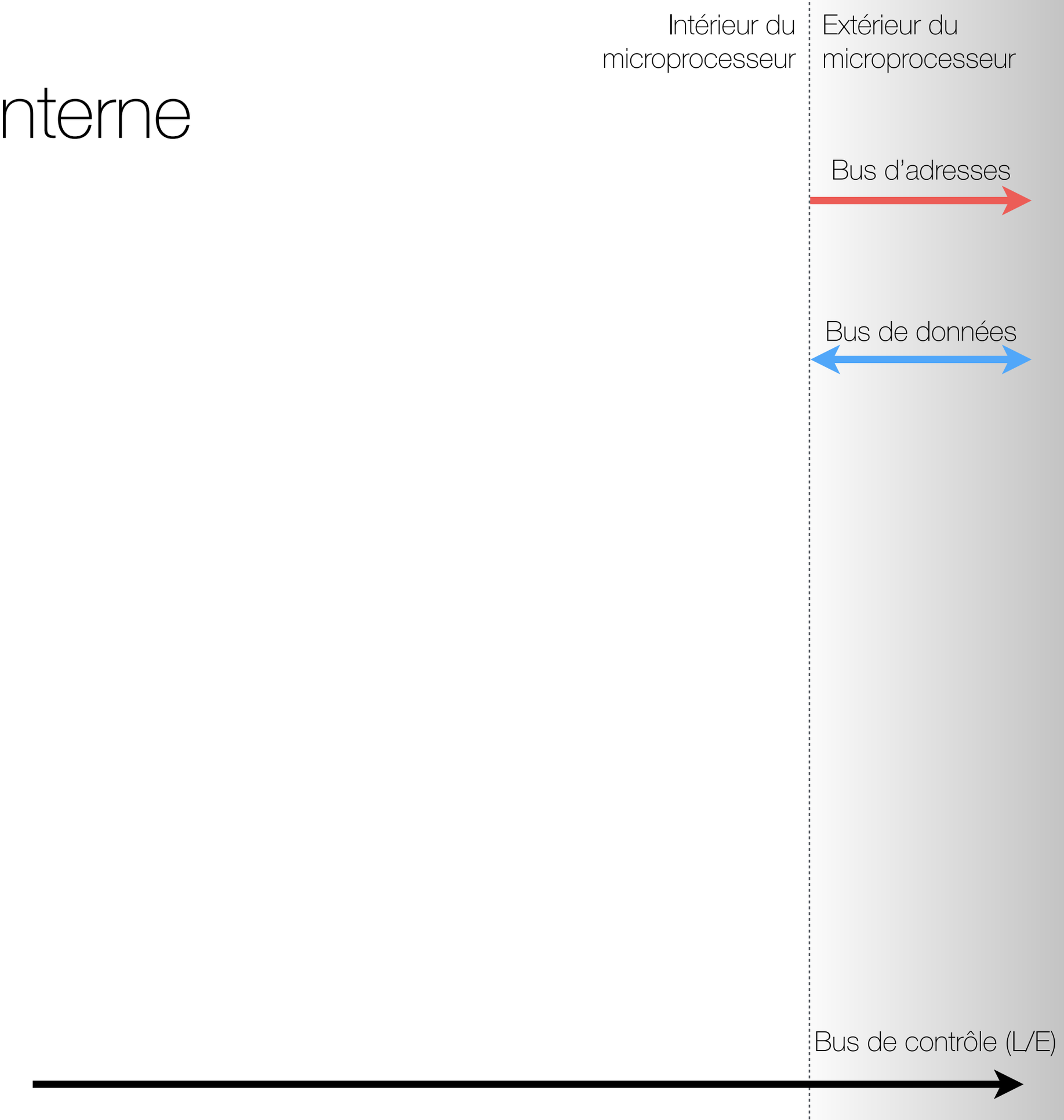
Cycle d'instructions

Que fait le microprocesseur?

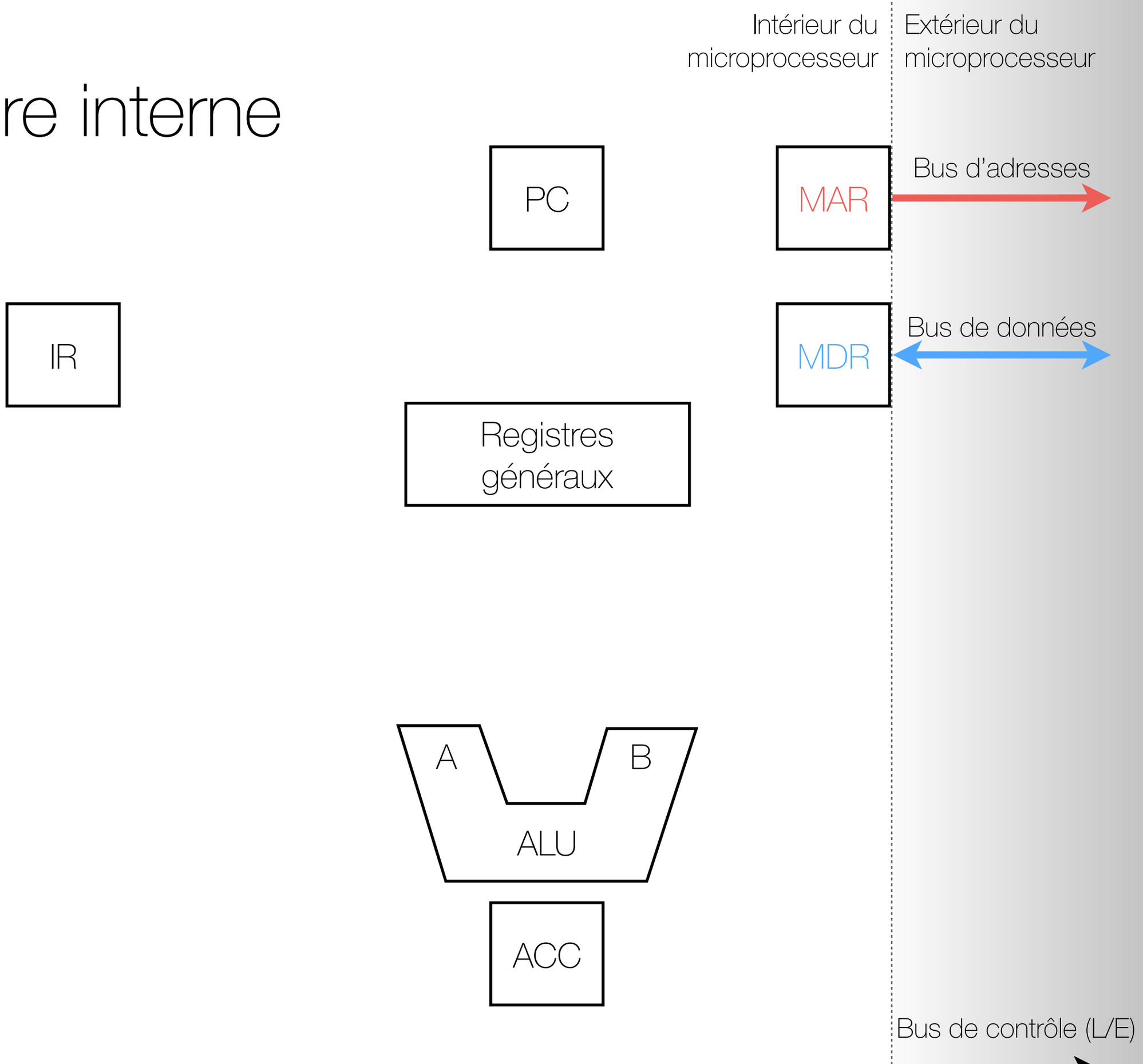
1. Lire: aller chercher la prochaine instruction
2. Décode: décode l'instruction (détermine ce qu'il y a à faire)
3. Exécute: exécuter l'instruction



Structure interne



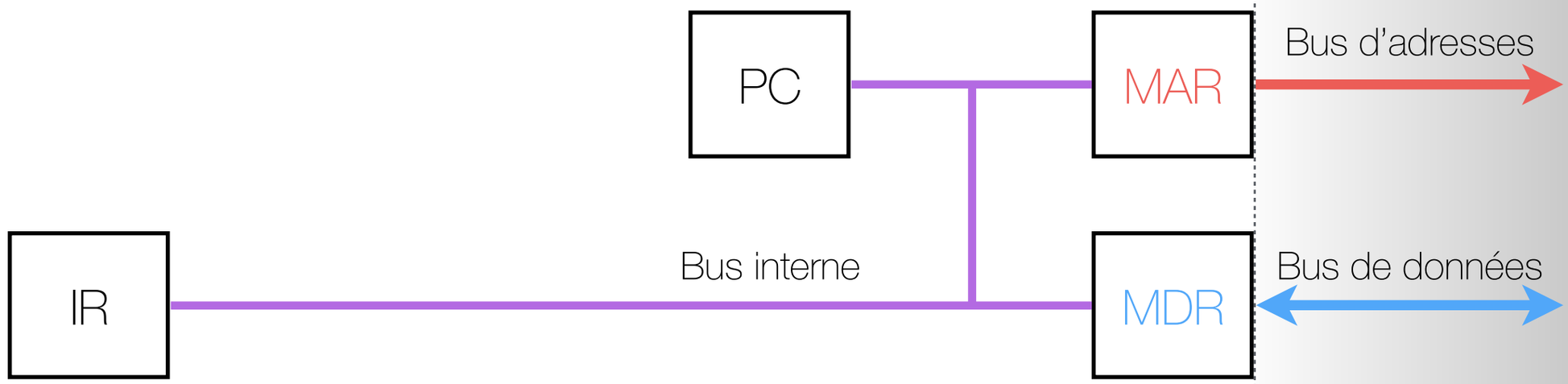
Structure interne



“Fetch” en détails…

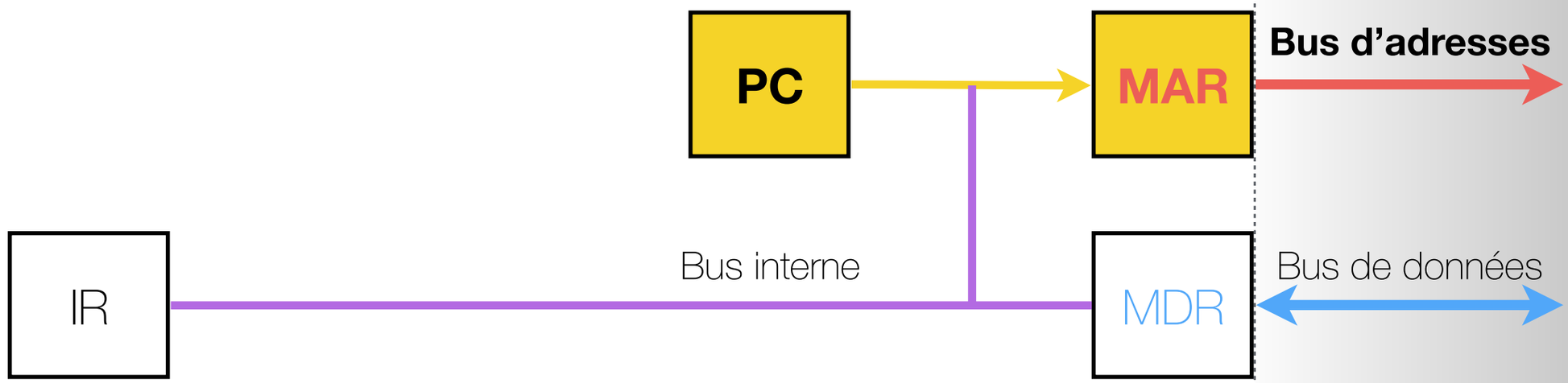
- Que se passe-t-il vraiment quand le micro-processeur veut lire la prochaine instruction?
- Où est l’instruction?
  - En mémoire, à l’adresse contenue dans registre PC (“Program Counter”)
- Où doit-on placer cette instruction?
  - Dans le registre IR (“Instruction Register”)

« Lire »



Action	Signification	µ-instructions
`Lecture`	`IR ← instruction`	`?`

« Lire »

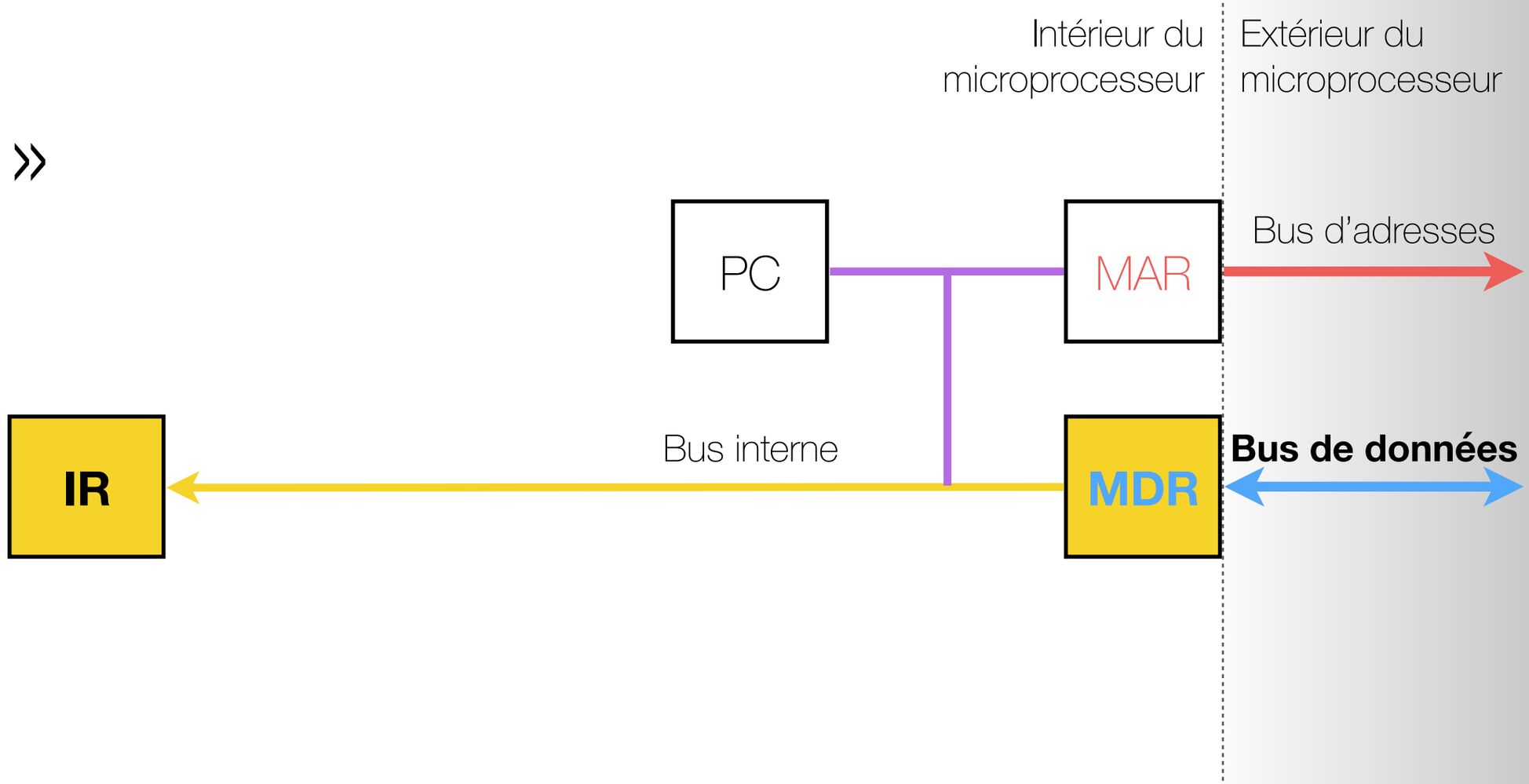


- MAR ← PC

- On place le contenu du PC dans le MAR
  - PC contient l'adresse de la prochaine exécution à exécuter.
  - On active le bus de contrôle en lecture
  - Après cette opération, l'exécution comme telle est disponible… dans le MDR

Bus de contrôle (**L**/E)

« Lire »



- IR ← MDR
  - On place le contenu du MDR dans l'IR
    - Voilà! l'instruction est dans l'IR, prête à être décodée

« Lire » en détails…

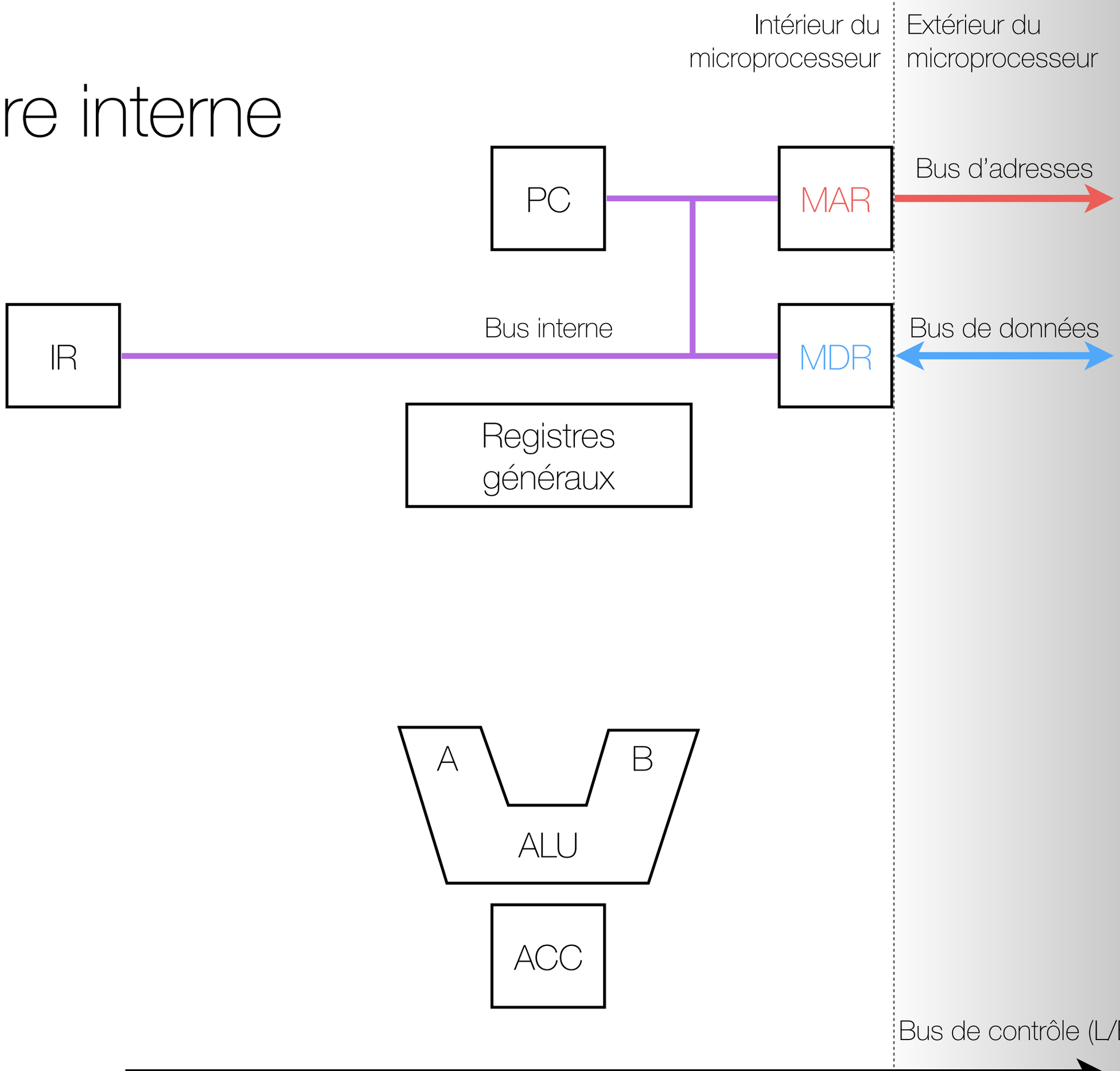
Action	Signification	µ-instructions
Lecture	IR ← instruction	MAR ← PC IR ← MDR

« Exécuter » en détails

- Admettons un moment que l'instruction dans IR ait été décodée. Comment faire pour exécuter l'instruction?

- Tout d'abord, il nous faut connecter les autres éléments de notre CPU sur le bus interne…

Structure interne



Intérieur du microprocesseur

Extérieur du microprocesseur

PC

MAR

Bus d'adresses

IR

Bus interne

MDR

Bus de données

Registres généraux

0

MUX A

MUX B

A

B

ALU

ACC

Bus de contrôle (L/E)

ALU

- Le microprocesseur en exemple a un ALU très simple. Cet ALU a:
  - deux entrées (A et B) et une sortie
  - deux « multiplexeurs » (MUX_A et MUX_B) qui déterminent quelles seront les entrées
- L'ALU peut être configuré pour effectuer différentes opérations (e.g. addition, soustraction).
- Le résultat de l'opération
  - se retrouve toujours dans l'accumulateur (ACC).
  - peut ensuite être propagé dans les registres de travail de l'ALU, sur le bus interne, ou à l'entrée (B) de l'ALU.
- C'est le CCU qui
  - détermine la configuration de l'ALU lors du décodage (voir plus loin)
  - détermine la configuration des MUX lors du décodage (voir plus loin)

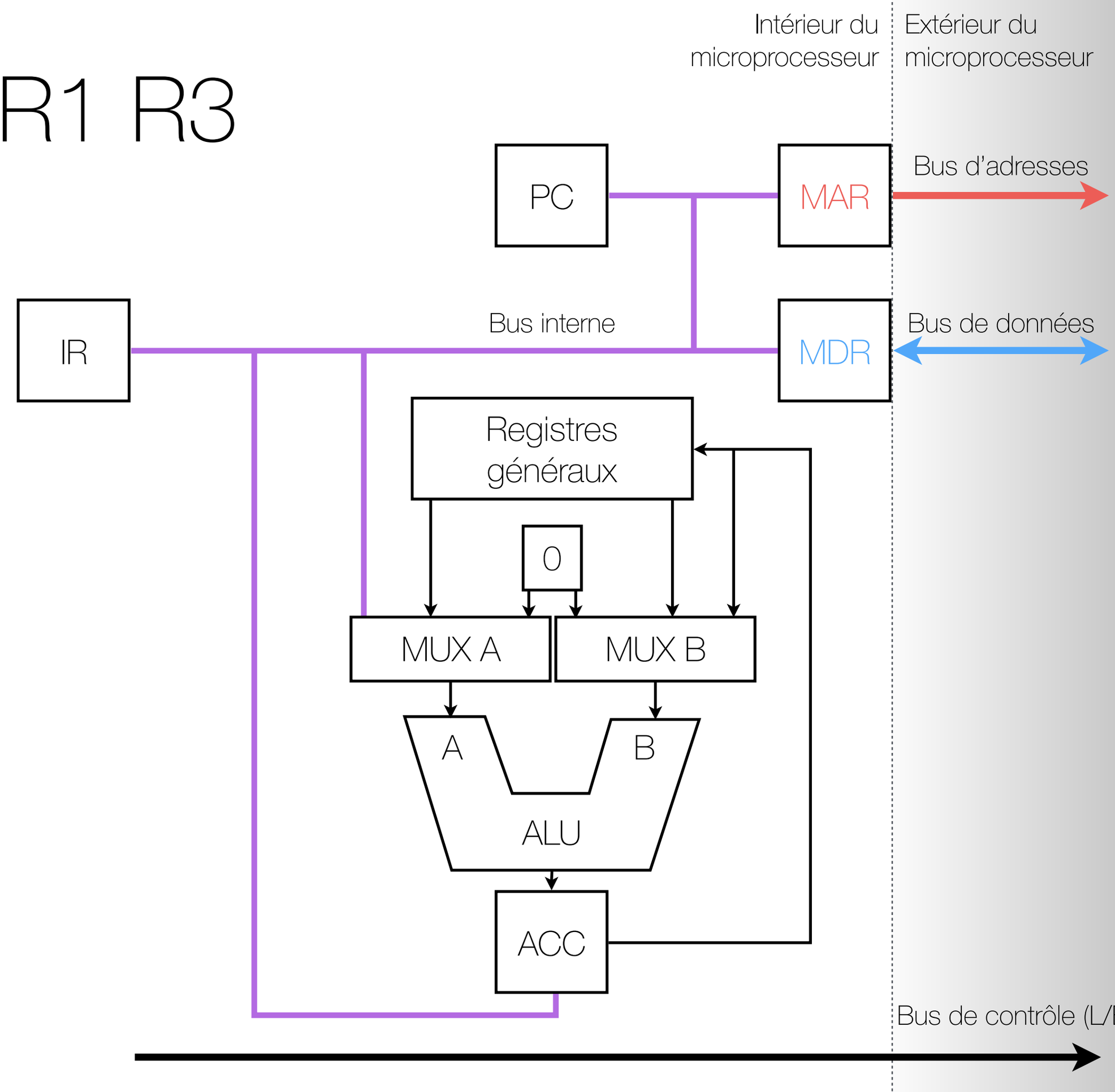
« Exécuter » en détails

- Maintenant que nous avons un ALU, son accumulateur et des registres généraux, nous pouvons voir comment un microprocesseur exécute ses instructions.
- Nous explorerons les instructions suivantes:
  - ADD R1 R3
  - MOV R0 #0x71
  - LDR R1 [R0]

ADD R1 R3

Instruction	Signification	µ-instructions
ADD R1 R3	R1 ← R1 + R3	?

ADD R1 R3


Intérieur du microprocesseur
Extérieur du microprocesseur
PC
MAR
Bus d'adresses
IR
Bus interne
MDR
Bus de données
Registres généraux
0
MUX A
MUX B
A
B
ALU
ACC
Bus de contrôle (L/E)

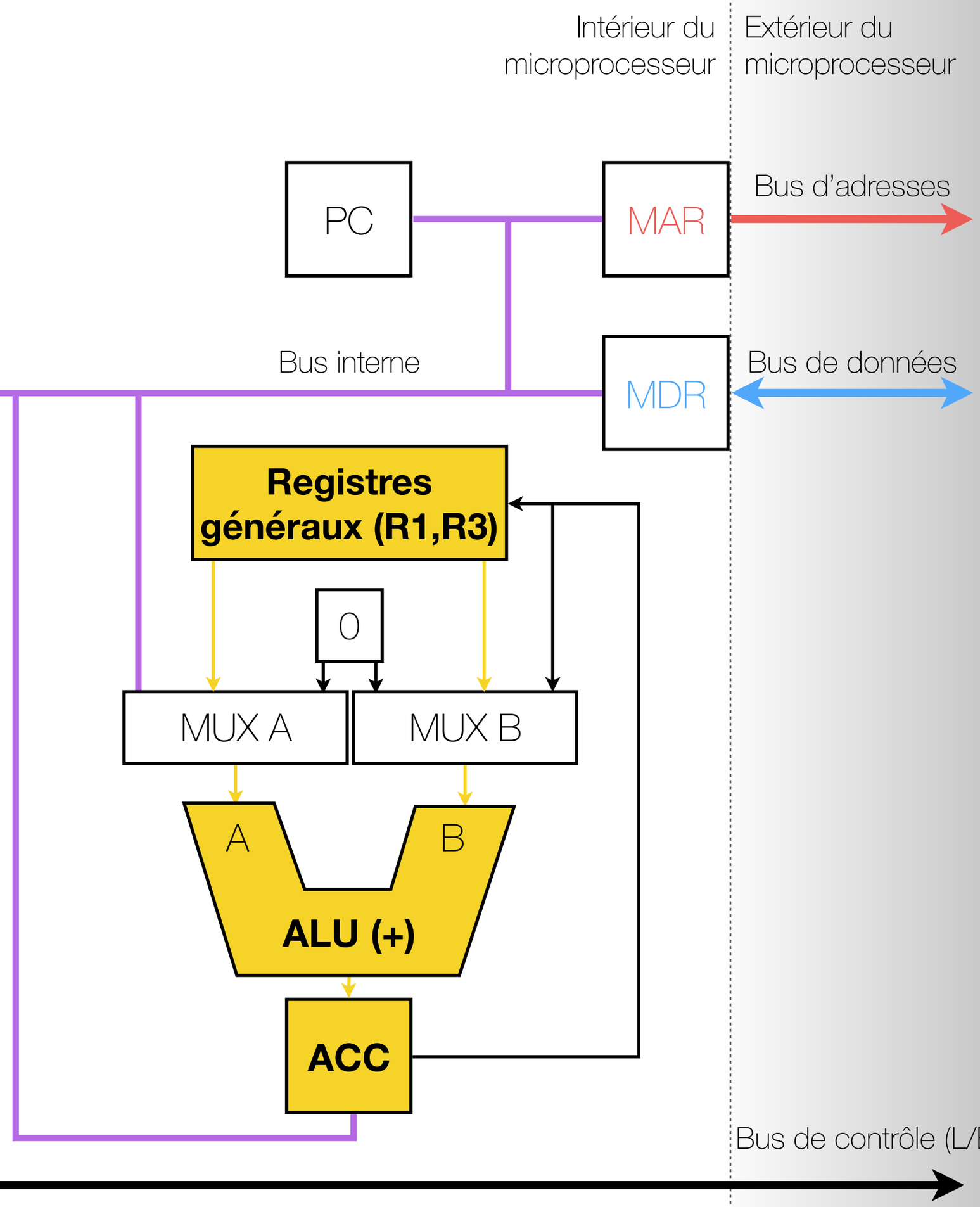
ADD R1 R3

- ACC ← R1 + R3

- On sélectionne R1 et R3 comme entrées de l'ALU

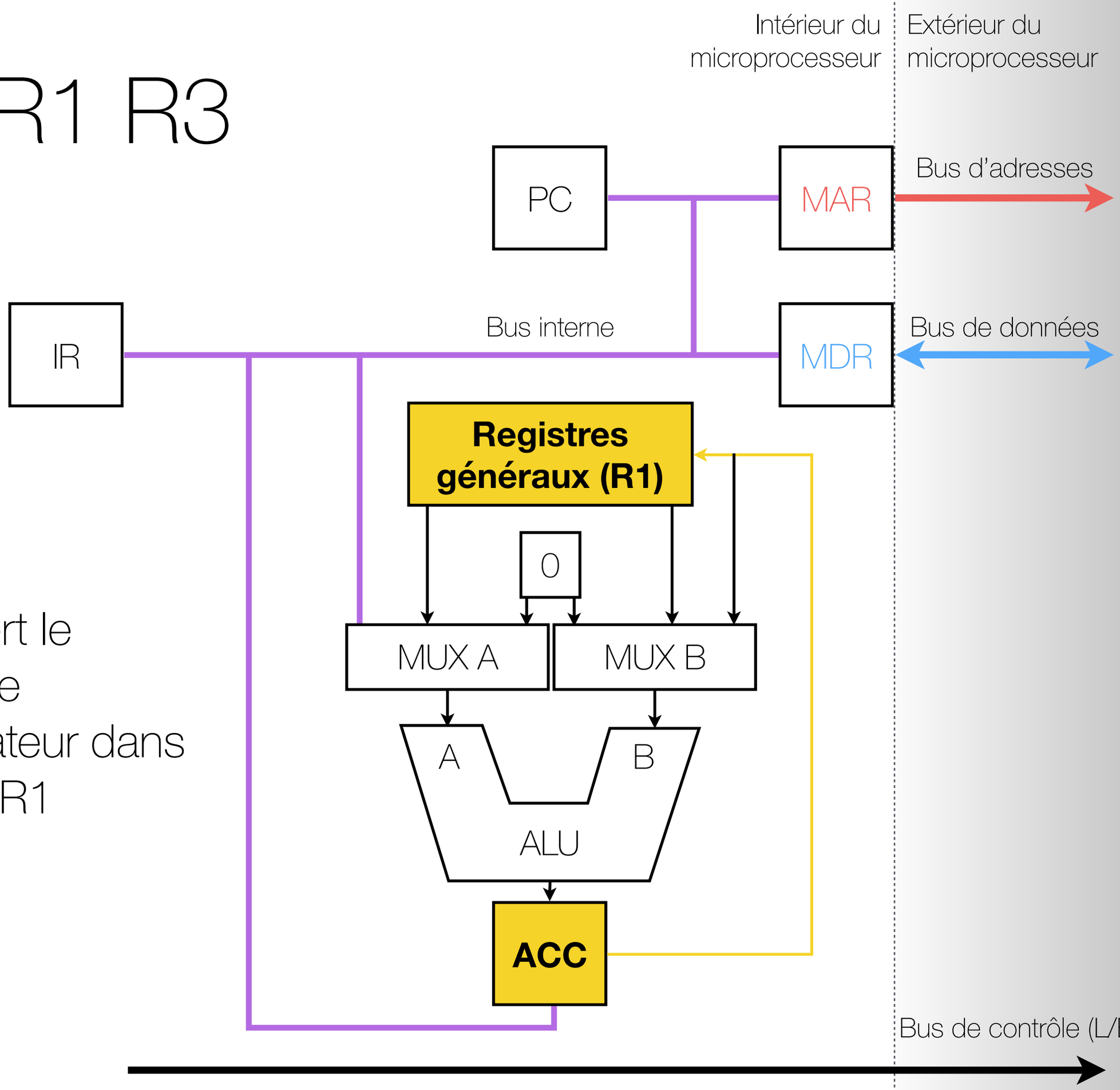
- L'ALU est configuré en addition

- Le résultat est placé dans l'accumulateur



ADD R1 R3

- R1 ← ACC
- On transfert le contenu de l’accumulateur dans le registre R1



ADD R1 R3

Instruction	Signification	μ-instructions
ADD R1 R3	R1 ← R1 + R3	ACC ← R1 + R3 R1 ← ACC

MOV R0 #0x71

Instruction	Signification	µ-instructions
MOV R0 #0x71	R0 ← 0x71	?

MOV R0 #0x71

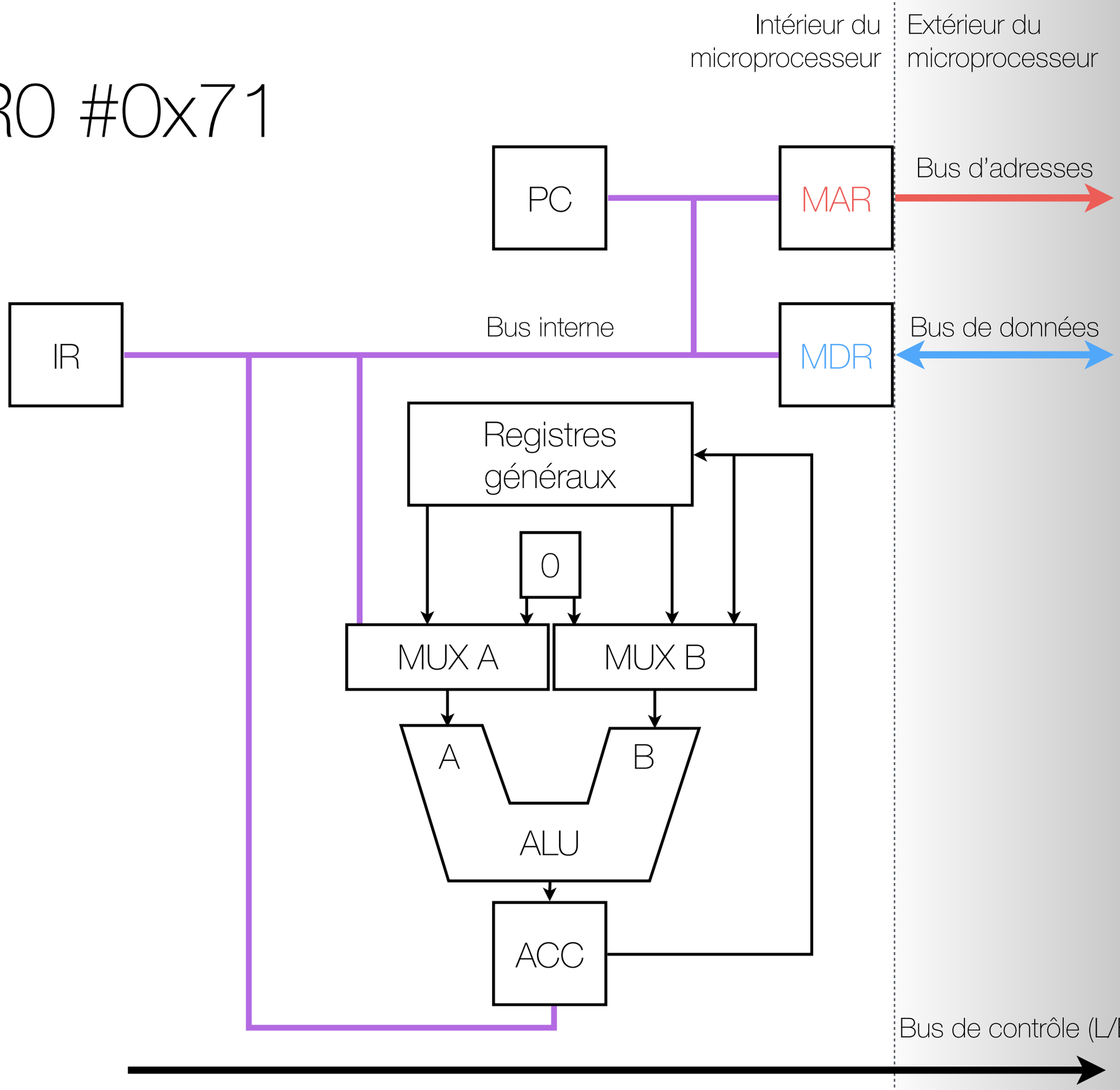
- Tout d'abord: quelle est la représentation binaire (sur 16 bits, comme dans le TP1) de cette instruction?

Toutes les instructions du microprocesseur sont sur 16 bits et se décomposent comme suit :
Bits 15 à 12 : Opcode de l'instruction
Bits 11 à 8 : Registre utilisé comme premier paramètre.
Bits 7 à 0 : Registre ou constante utilisés comme deuxième paramètre

Mnémonique	**Opcode**	**Description**
MOV Rd Rs	0000	Écriture de la valeur du registre Rs dans le registre Rd
MOV Rd Const	0100	Écriture d'une constante dans le registre Rd

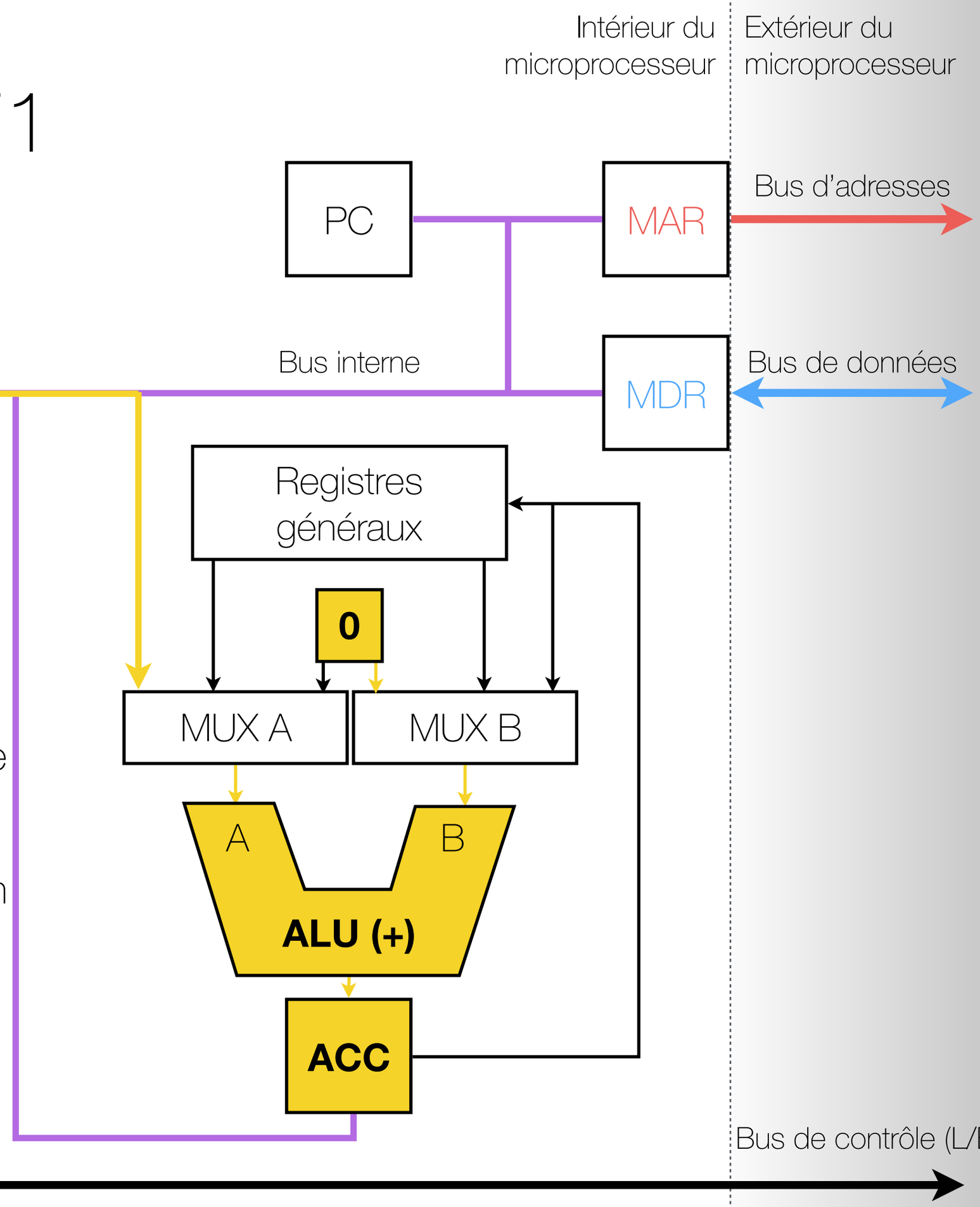
Instruction	Signification	Binaire (16 bits)
`MOV R0 #0x71`	`R0 ← 0x71`	`0x4071`

MOV R0 #0x71


Intérieur du microprocesseur
Extérieur du microprocesseur
PC
MAR
Bus d'adresses
IR
Bus interne
MDR
Bus de données
Registres généraux
0
MUX A
MUX B
A
B
ALU
ACC
Bus de contrôle (L/E)

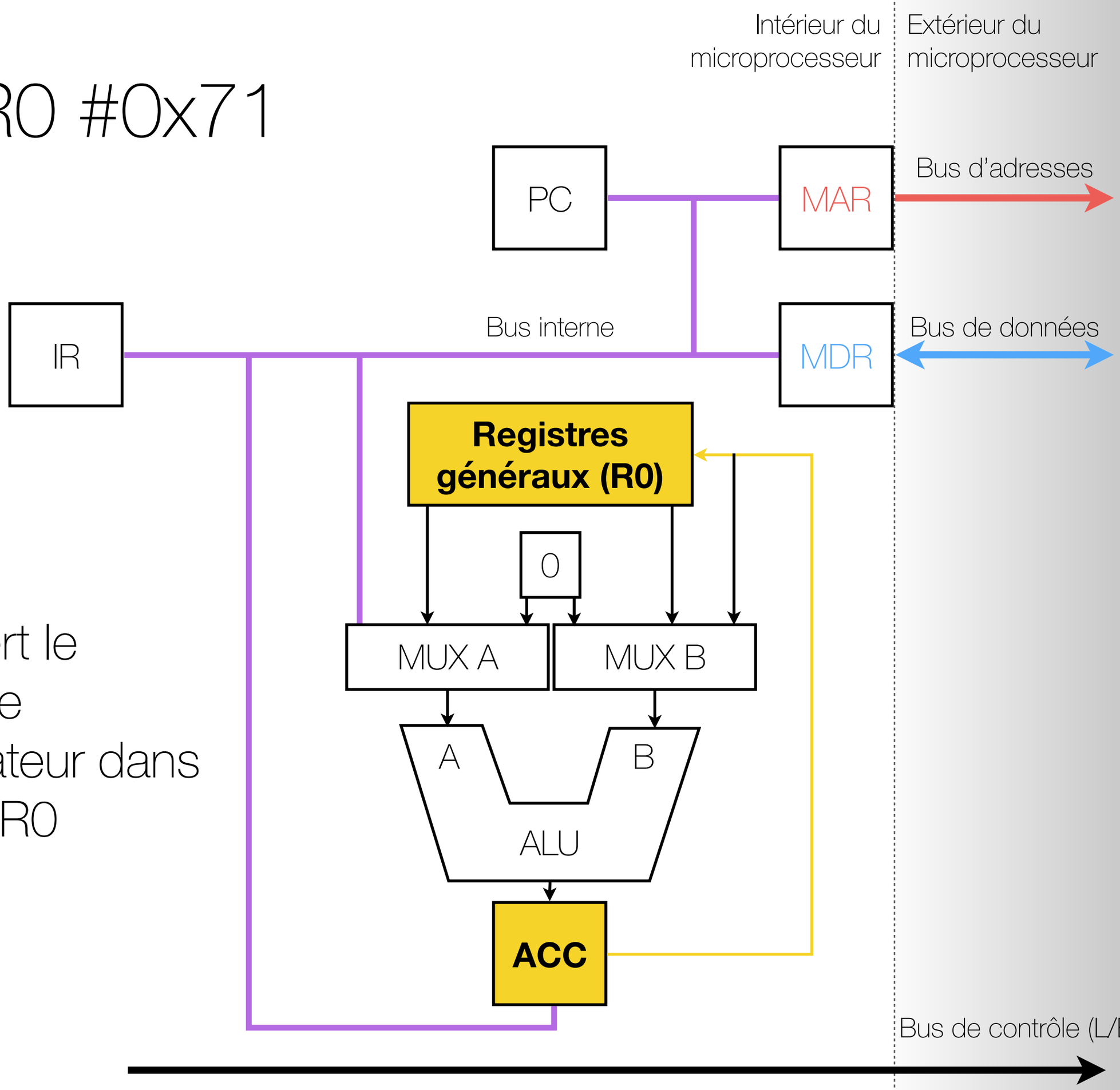
MOV R0 #0x71

- ACC ← 0 + IR[#0x71]
  - On place la portion de l'instruction qui correspond à #0x71 à l'ALU
  - On place '0' à l'autre entrée de l'ALU
  - L'ALU est configuré en addition
  - Le résultat (0+0x71) est stocké dans l'accumulateur



MOV R0 #0x71

- R0 ← ACC
  - On transfert le contenu de l'accumulateur dans le registre R0



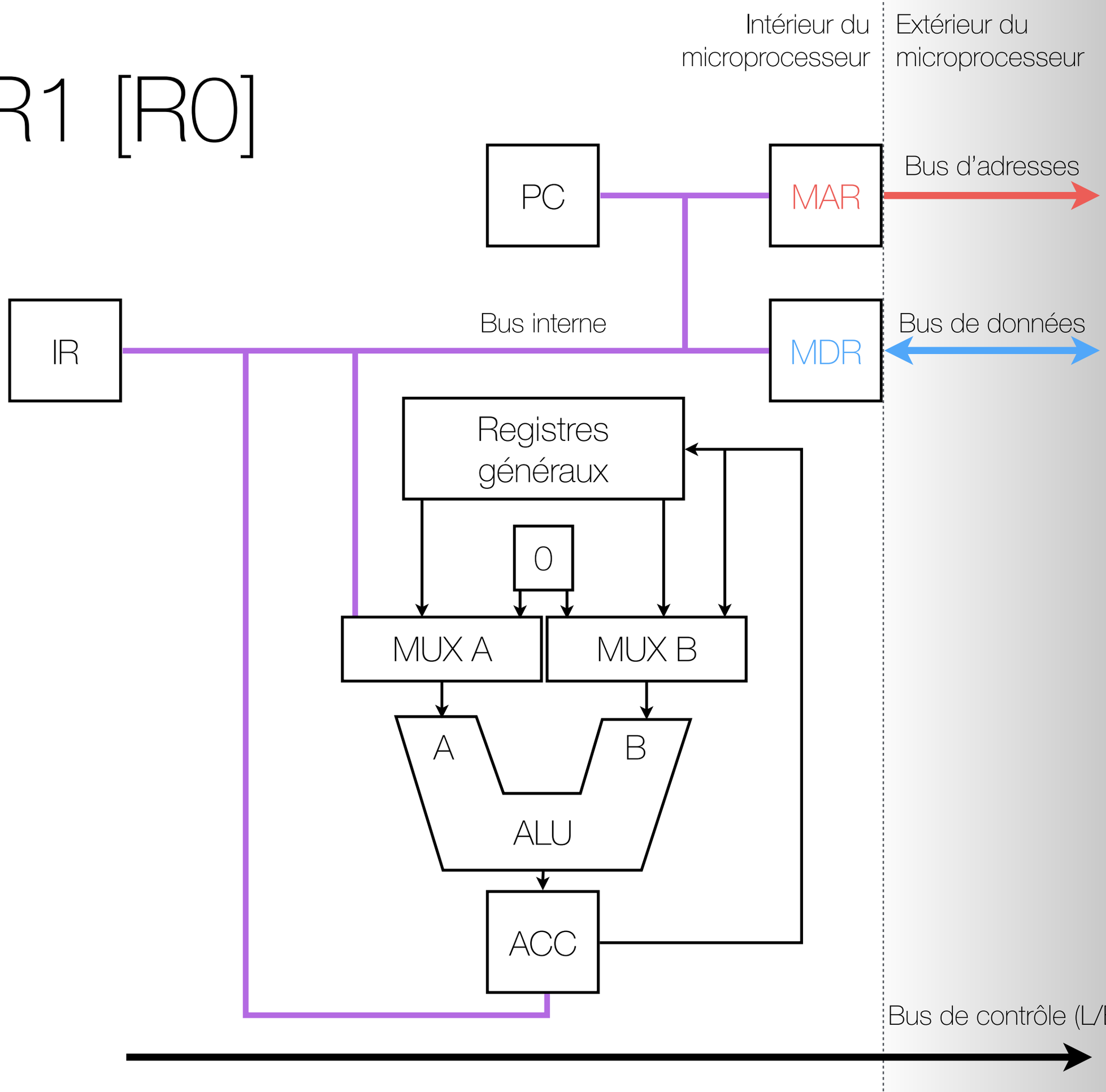
MOV R0 #0x71

Instruction	Signification	µ-instructions
MOV R0 #0x71	R0 ← 0x71	ACC ← 0 + IR[#0x71] R0 ← ACC

LDR R1 [R0]

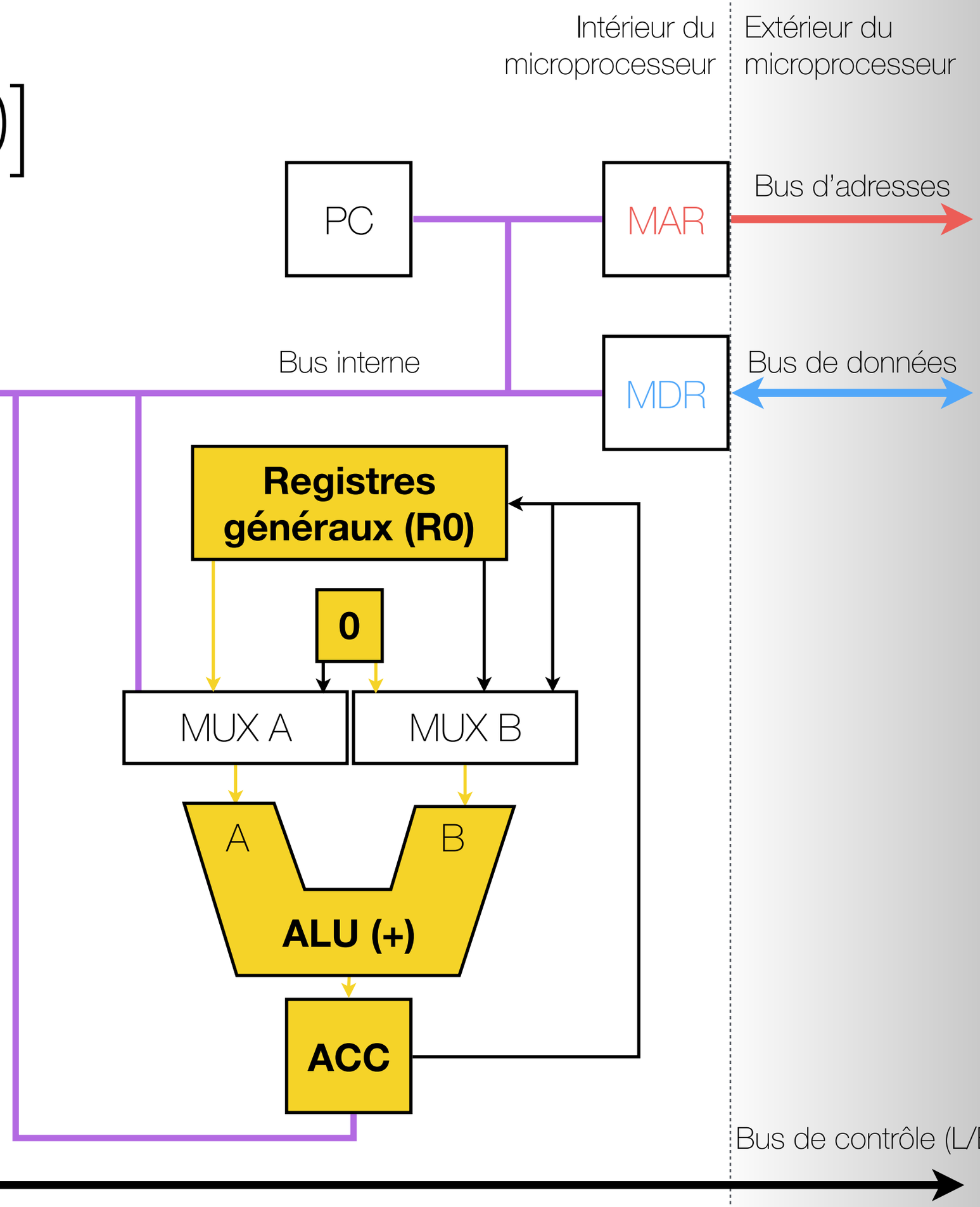
Instruction	Signification	µ-instructions
`LDR R1 [R0]`	`R1 ← Memoire[R0]`	?

LDR R1 [R0]


Intérieur du microprocesseur
Extérieur du microprocesseur
PC
MAR
Bus d'adresses
IR
Bus interne
MDR
Bus de données
Registres généraux
0
MUX A
MUX B
A
B
ALU
ACC
Bus de contrôle (L/E)

LDR R1 [R0]

- ACC ← 0 + R0
  - On place R0 à l'entrée de l'ALU
  - On place '0' à l'autre entrée de l'ALU
  - L'ALU est configuré en addition
  - Le résultat (0+R0) est stocké dans l'accumulateur

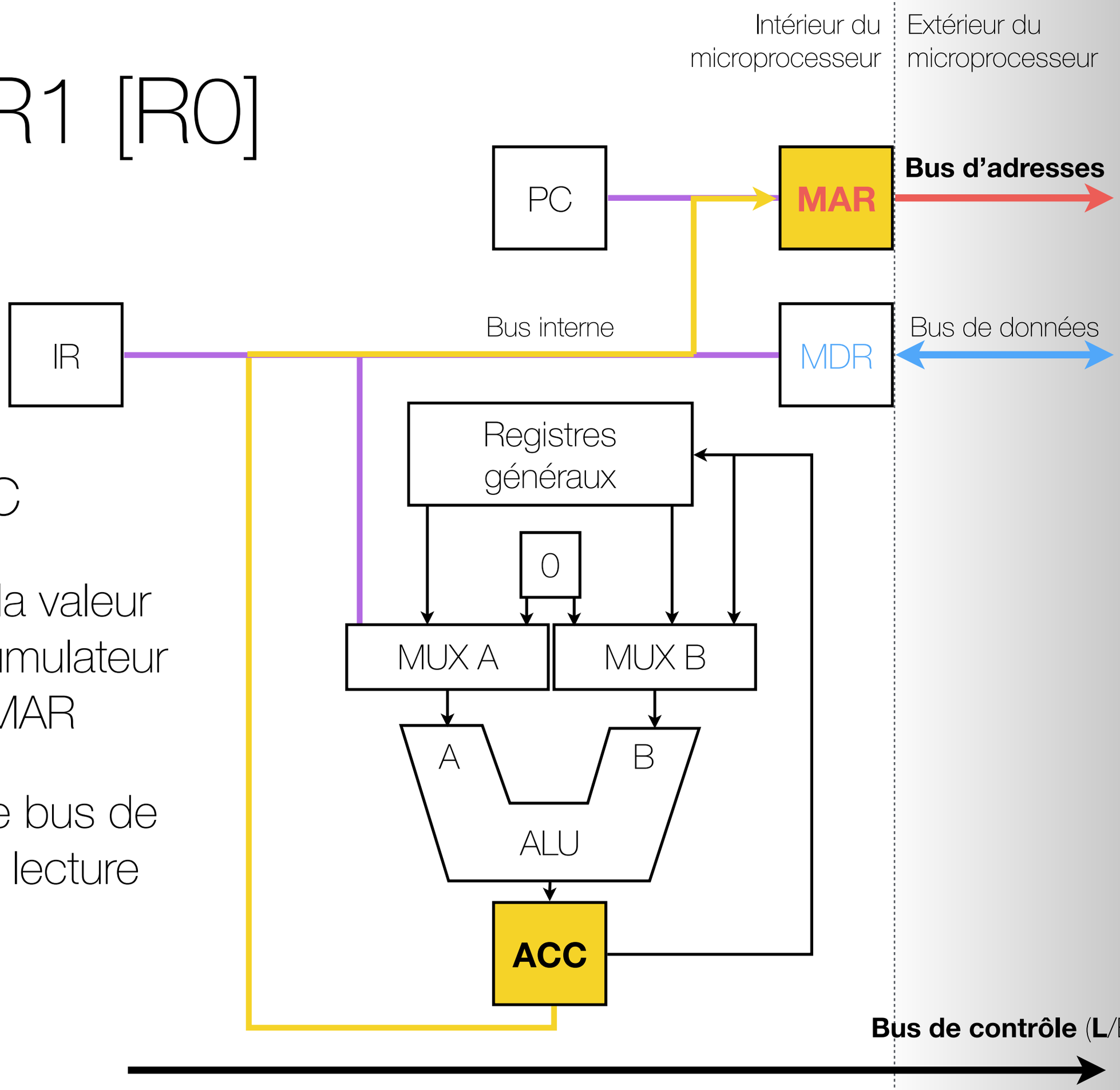


LDR R1 [R0]

- MAR ← ACC

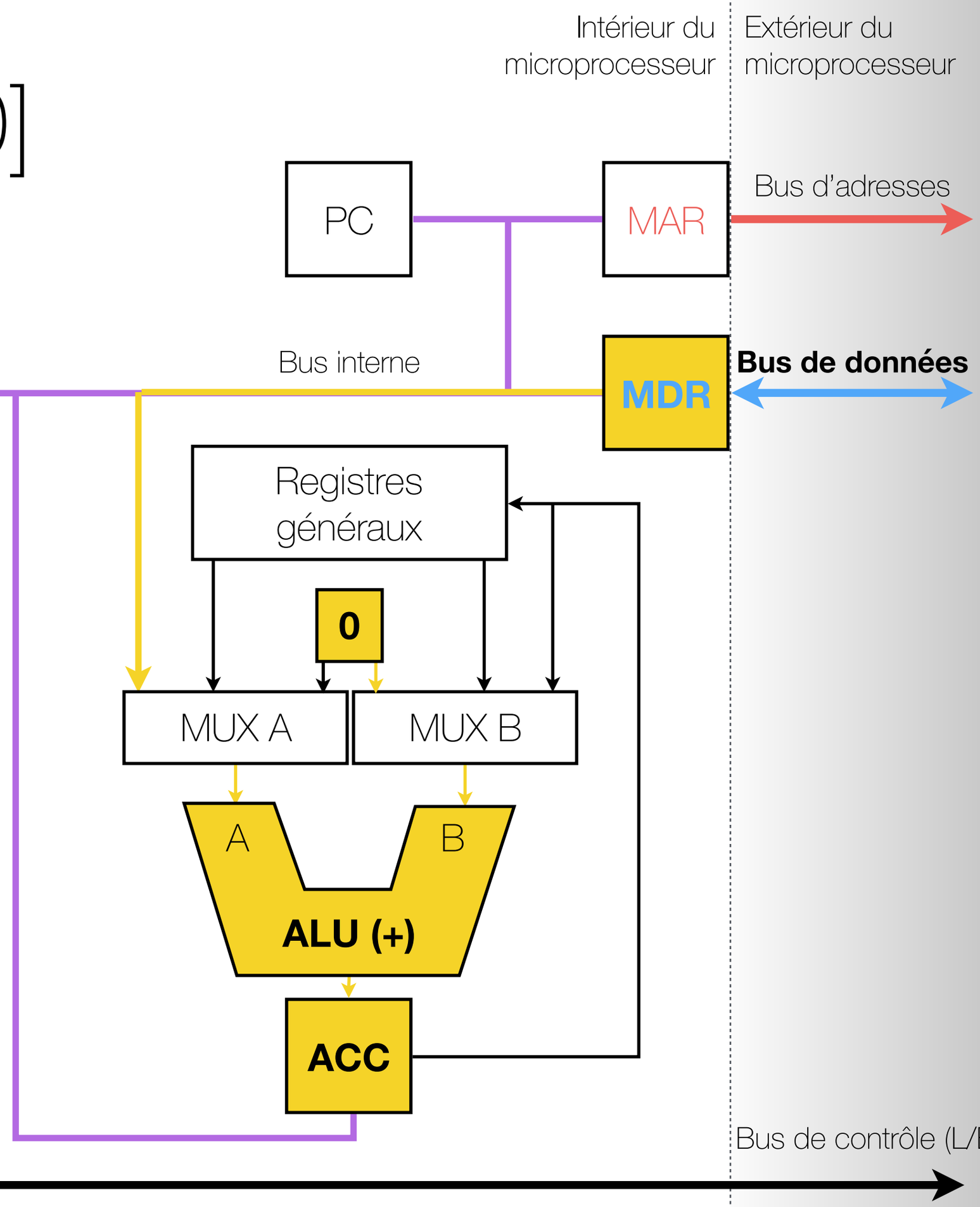
- On stocke la valeur dans l'accumulateur (R0) dans MAR

- On active le bus de contrôle en lecture



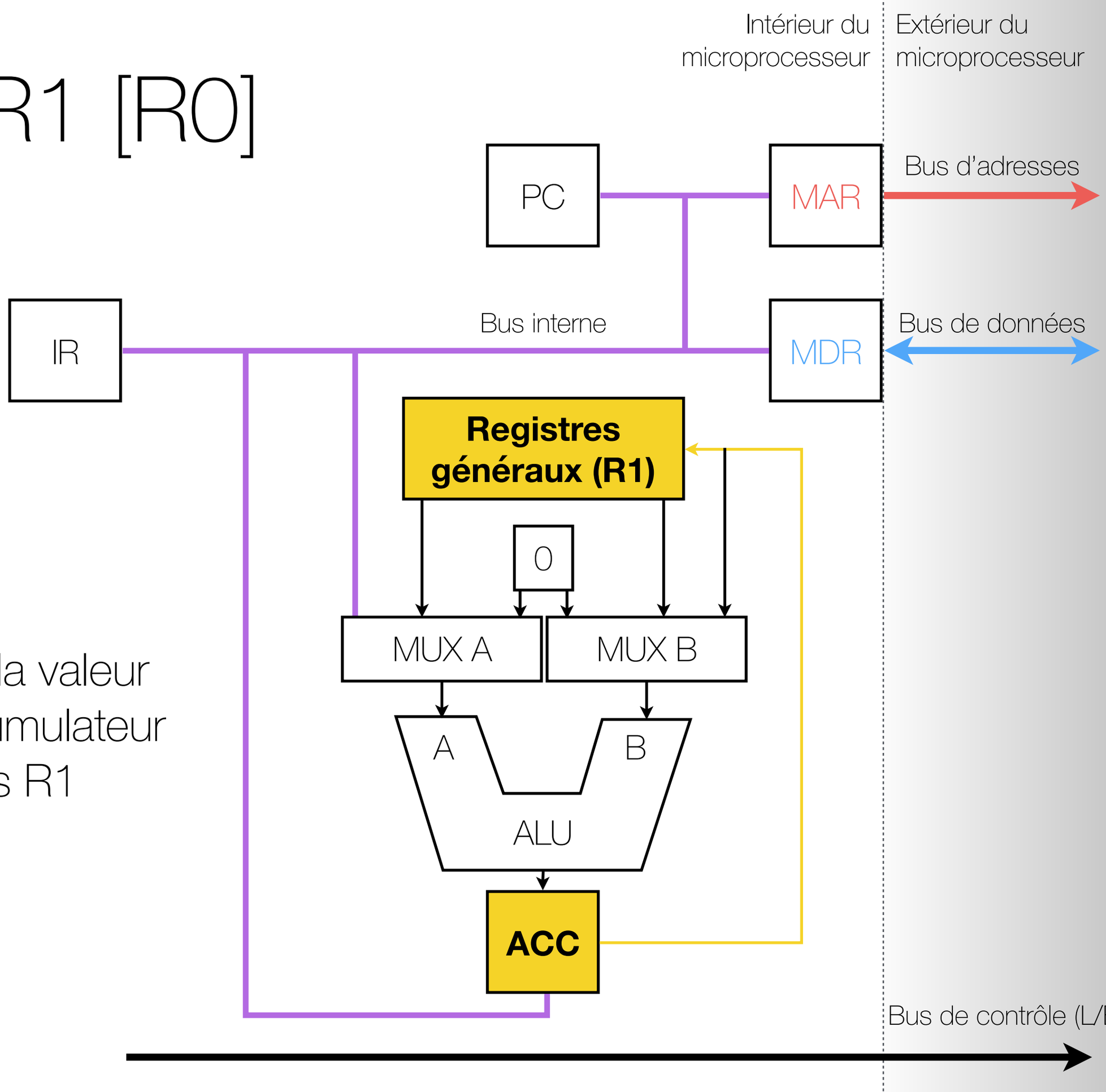
LDR R1 [R0]

- ACC ← 0 + MDR
  - On place MDR à l'entrée de l'ALU
  - On place '0' à l'autre entrée de l'ALU
  - L'ALU est configuré en addition
  - Le résultat (0+MDR) est stocké dans l'accumulateur



LDR R1 [R0]

- R1 ← ACC
  - On stocke la valeur dans l'accumulateur (MDR) dans R1



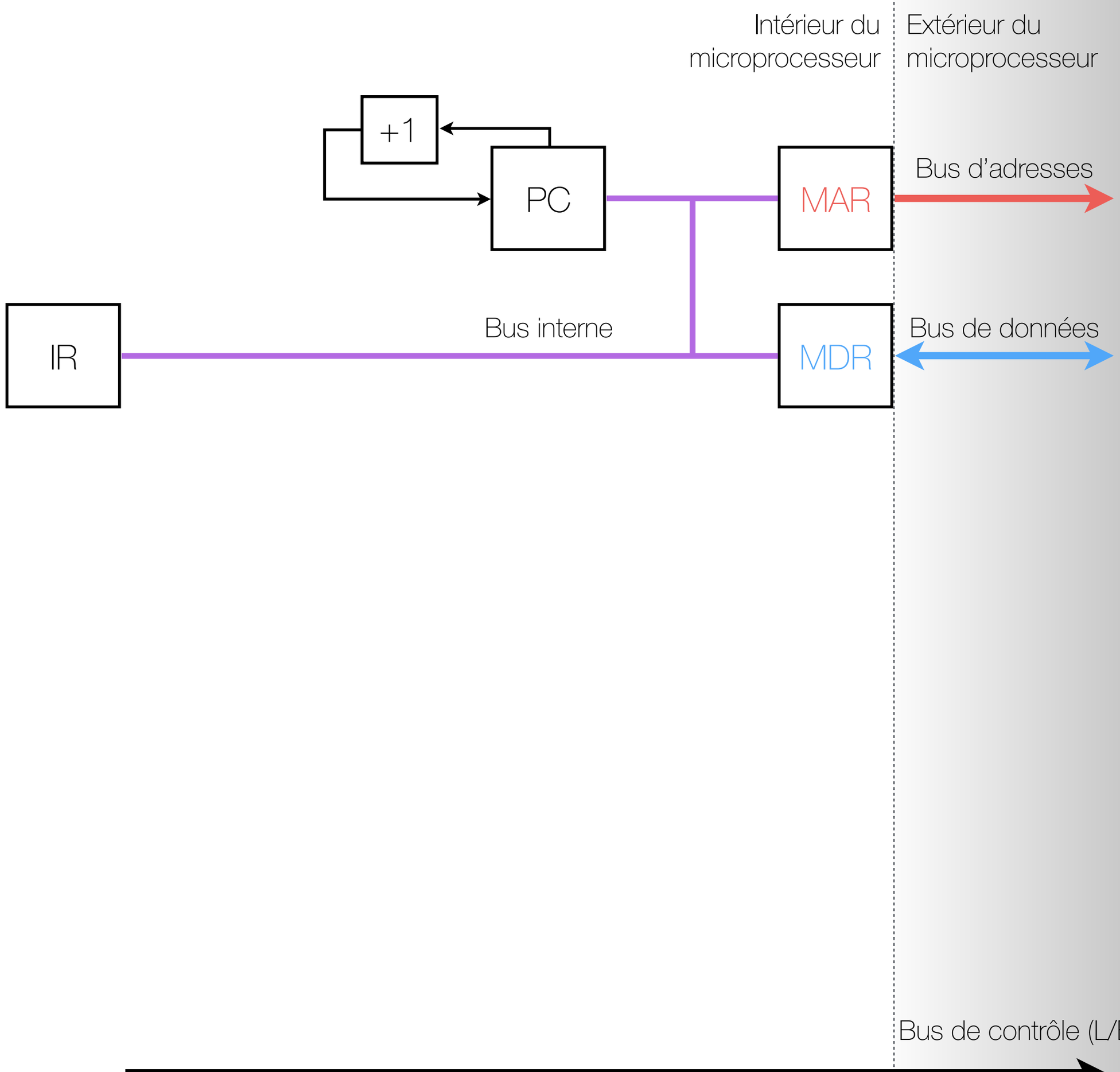
LDR R1 [R0]

Instruction	Signification	µ-instructions
LDR R1 [R0]	R1 ← Memoire[R0]	ACC ← R0 + 0 MAR ← ACC ACC ← MDR + 0 R1 ← ACC

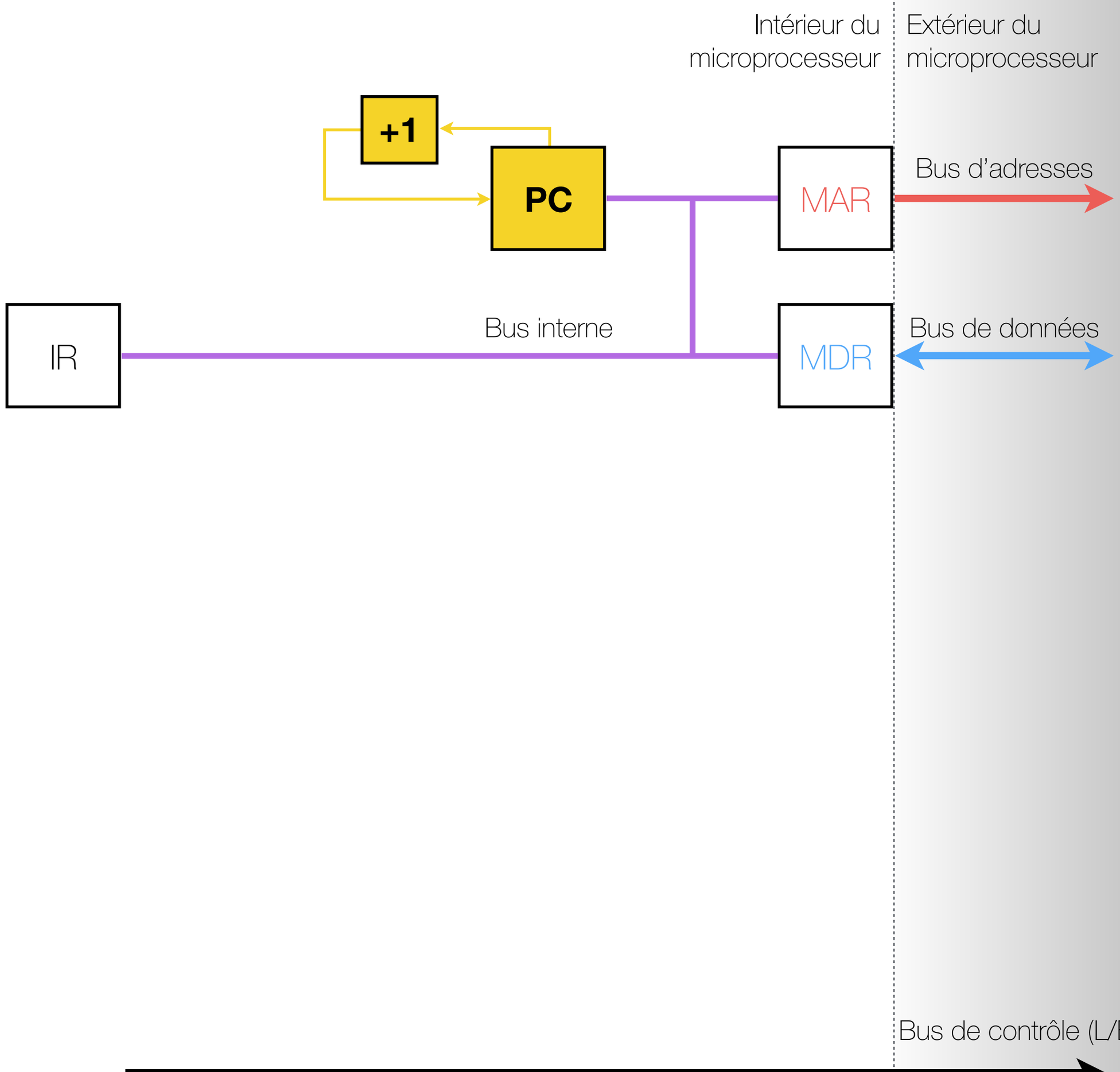
Après chaque exécution…

Action	μ-instructions
Incrémenter PC	PC ← PC + 1

PC


Intérieur du microprocesseur
Extérieur du microprocesseur
+1
PC
MAR
Bus d'adresses
IR
Bus interne
MDR
Bus de données
Bus de contrôle (L/E)

PC

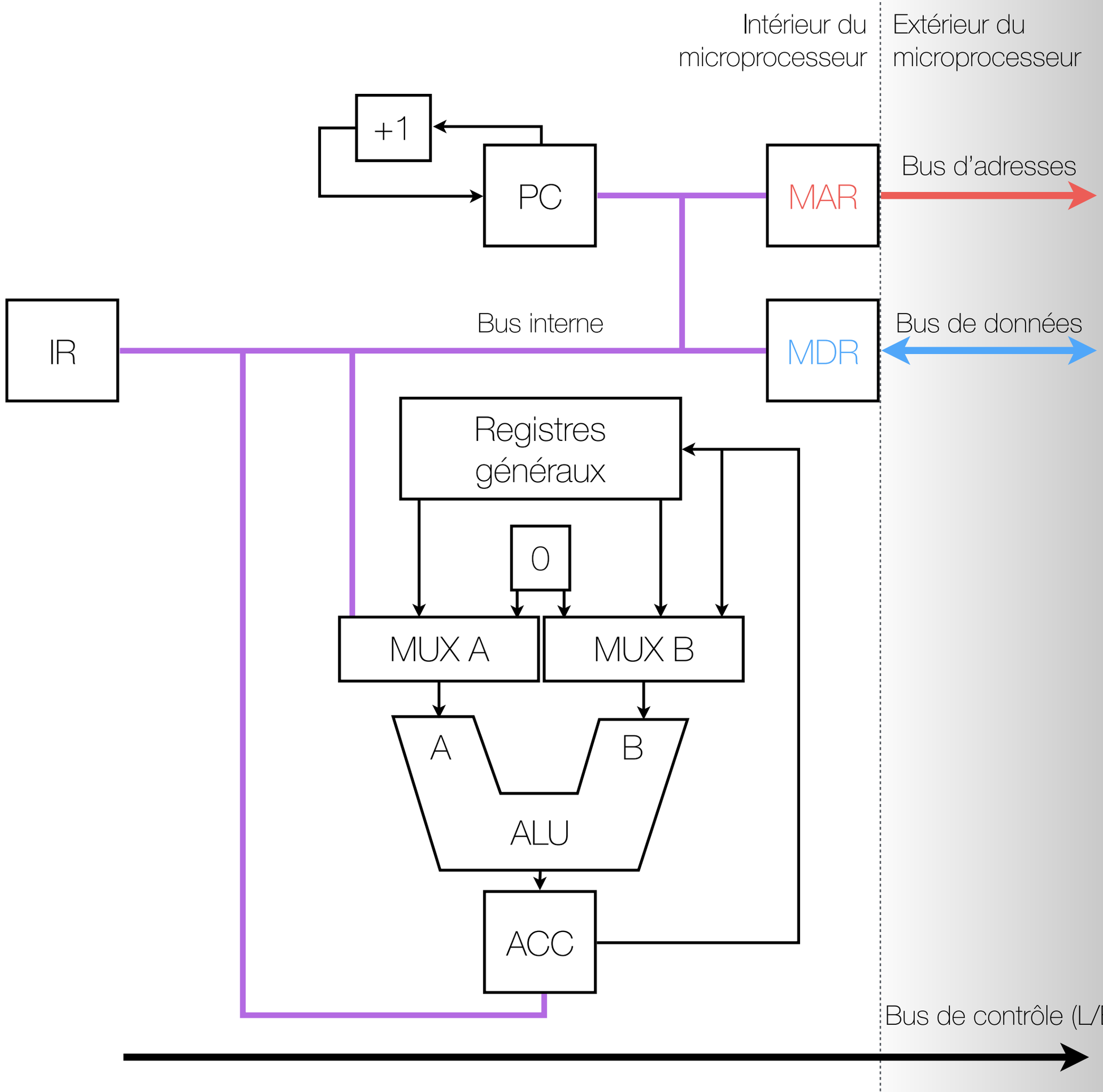

Intérieur du microprocesseur
Extérieur du microprocesseur
+1
PC
MAR
Bus d'adresses
IR
Bus interne
MDR
Bus de données
Bus de contrôle (L/E)

Résumé (lecture & exécution)

Instruction	Étapes	μ-instructions
ADD R1 R3	lecture	MAR ← PC IR ← MDR
	exécution	ACC ← R1 + R3 R1 ← ACC
	incrémente PC	PC ← PC+1
MOV R0 #0x71	lecture	MAR ← PC IR ← MDR
	exécution	ACC ← 0 + IR[#0x71] R0 ← ACC
	incrémente PC	PC ← PC+1

Le « décodage » en détails…

- Nous avons vu comment la prochaine instruction est lue (“fetch”) et exécutée (“execute”), mais comment est-elle décodée?
    - En d’autres mots, comment le microprocesseur parvient à traduire l’instruction dans IR en une séquence de micro-instructions?
- 3 composantes:
    - Mémoire de décodage
    - Compteur
    - Mémoire de µ-instructions


Intérieur du microprocesseur
Extérieur du microprocesseur
+1
PC
MAR
Bus d'adresses
IR
Bus interne
MDR
Bus de données
Registres généraux
0
MUX A
MUX B
A
B
ALU
ACC
Bus de contrôle (L/E)

Intérieur du microprocesseur

Extérieur du microprocesseur

+1

PC

MAR

Bus d'adresses

IR

Bus interne

MDR

Bus de données

Mémoire décodage

Registres généraux

0

Compteur

MUX A

MUX B

Mémoire µ-instructions

A

B

ALU

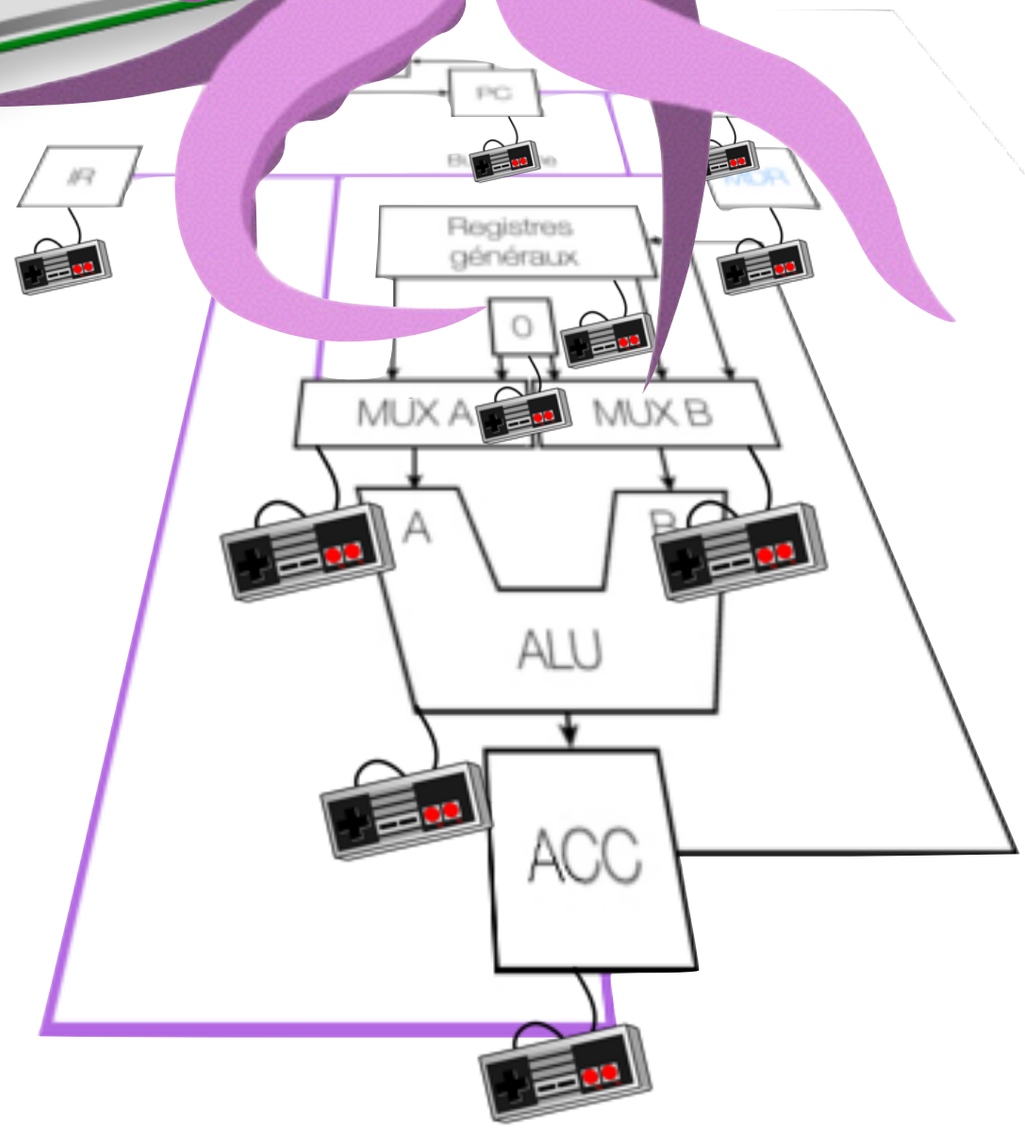
ACC

Bus de contrôle (L/E)

PC sur bus interne
MAR lit le bus interne
…

1. Mets PC dans MAR
2. Mets MDR dans IR
3. …

Mémoire de
μ-instructions



Mémoire de
décodage

Instruction

0x4071

Nous!

Décodage

- Mémoire de décodage
  - détermine l'emplacement de la première µ-instruction à effectuer pour effectuer l'instruction.
    - Entrée: l'opcode de l'instruction
    - Sortie: Numéro de µ-instruction à exécuter
- Compteur
  - détermine quelle sera la prochaine µ-instruction à faire.
    - La valeur initiale de ce compteur provient de l'opcode de l'instruction via la mémoire de décodage.
    - Par la suite, le compteur est incrémenté à chaque coup d'horloge, changeant la µ-instruction en cours.

Opcode	Adresse
0x0	0x0
MOV Rd #cte	0x2
…	…

Compteur

Adresse	*µ*-instruction
0x0	MAR ← PC
0x1	IR ← MDR
0x2	ACC ← 0 + IR[#cte]
0x3	R0 ← ACC
0x4	PC ← PC + 1
0x5	MAR ← PC
0x6	IR ← MDR
0x7	…

Décodage

- Mémoire de µ-instructions
  - mémoire contrôlant l'ensemble du microprocesseur.
    - Entrée: numéro de micro-instruction
    - Sortie: valeurs (prédéterminées) pour toutes les lignes de contrôle du microprocesseur
  - cette mémoire détermine:
    - si les registres sont chargés ou lus
    - quelle sera l'opération effectuée par l'ALU;
    - quelle sera la valeur de certaines lignes du bus de contrôle. Elles partent toutes du CCU et elles se rendent au diverses composantes du microprocesseur.

Opcode	Adresse
0x0	0x0
MOV Rd #cte	0x2
…	…

Compteur

Adresse	*µ*-instruction
0x0	MAR ← PC
0x1	IR ← MDR
0x2	ACC ← 0 + IR[#cte]
0x3	R0 ← ACC
0x4	PC ← PC + 1
0x5	MAR ← PC
0x6	IR ← MDR
0x7	…

Décodage

Adresse	*μ*-instruction
0x0	MAR ← PC
0x1	IR ← MDR
0x2	ACC ← 0 + IR[#cte]
0x3	R0 ← ACC
0x4	PC ← PC + 1
0x5	MAR ← PC
0x6	IR ← MDR
0x7	…

Signaux de contrôle
PC en écriture sur le bus interne
MAR en lecture sur le bus interne

Signaux de contrôle
8 derniers bits de IR sur le bus interne
MUX A sélectionne bus interne
MUX B sélectionne «0»
ALU en addition

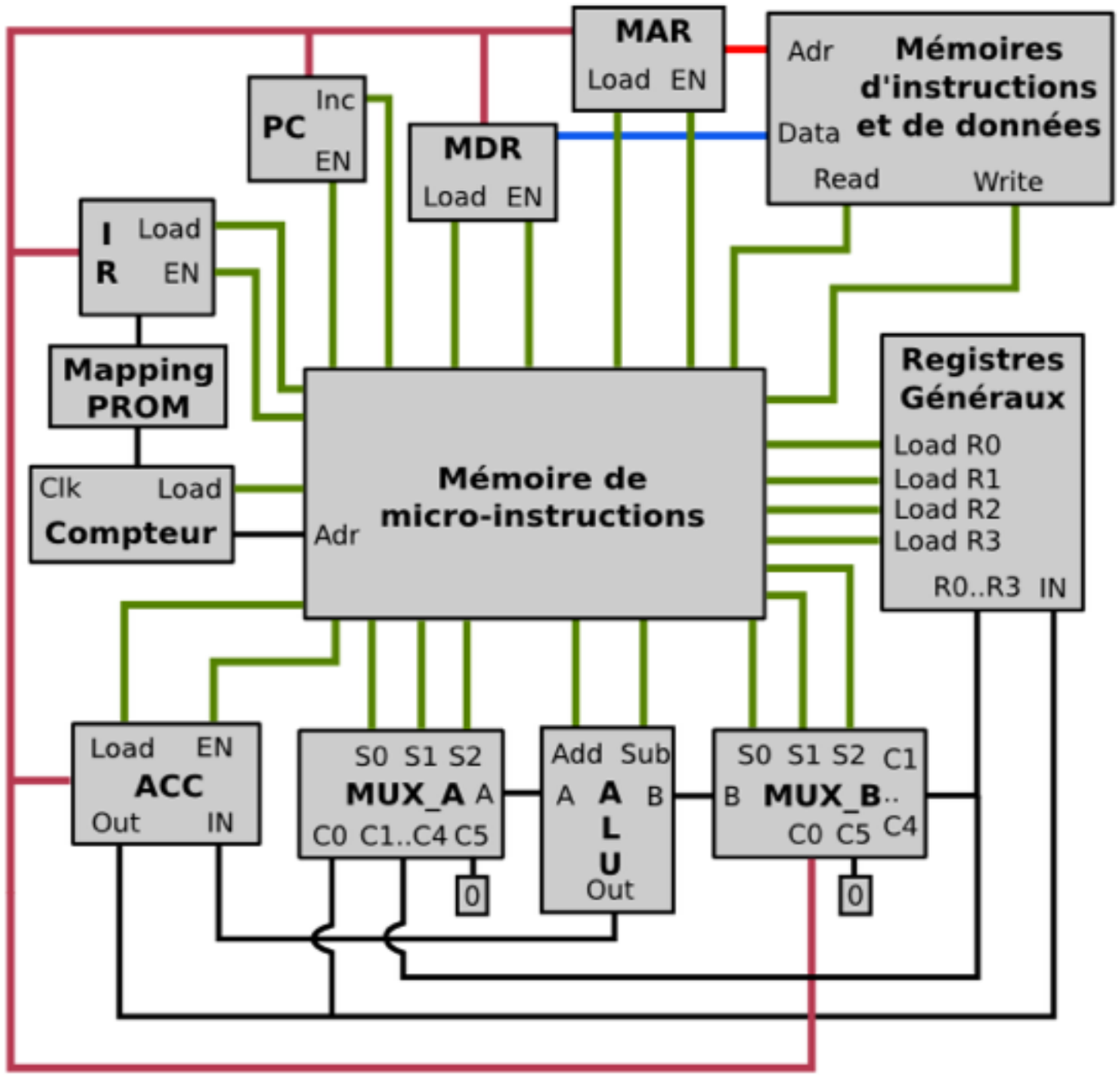
Signaux de contrôle
Active PC+1

Exercice

- Avec les instructions ci-bas, écrivez un programme qui additionne deux nombres:
  - Le premier est à l'adresse mémoire #0x71
  - Le deuxième est à l'adresse mémoire #0x72
  - Le résultat est sauvegardé en mémoire, à l'adresse #0x73

Mnémonique	**Opcode**	**Description**
MOV Rd Rs	0000	Écriture de la valeur du registre Rs dans le registre Rd
MOV Rd Const	0100	Écriture d'une constante dans le registre Rd
ADD Rd Rs	0001	Addition des valeurs des registres Rd et Rs et insertion du résultat dans le registre Rd
ADD Rd Const	0101	Addition de la valeur du registre Rd avec une constante et insertion du résultat dans Rd
SUB Rd Rs	0010	Soustraction de la valeur Rs à l'intérieur de registre Rd.
SUB Rd Const	0110	Soustraction d'une constante à l'intérieur du registre Rd
LDR Rd [Rs]	1000	Chargement d'une valeur se trouvant à l'adresse Rs de l'ordinateur dans un registre.
STR Rd [Rs]	1001	Écriture de la valeur d'un registre à l'adresse Rs de l'ordinateur.

Démonstration sur simulateur



Programmes, instructions, et micro-instructions

Programme

```
c = a+b;
```

Instructions

```
MOV R0 #0x71
LDR R1 [R0]
MOV R0 #0x72
LDR R2 [R0]
ADD R1 R2
…
```

Micro-instructions

```
0 + IR[#0x71]
R0 ← ACC
PC ← PC+1
MAR ← PC
IR ← MDR
```

Micro-instructions

```
ACC ← R0 + 0
MAR ← ACC
ACC ← 0 + MDR
R1 ← ACC
PC ← PC+1
MAR ← PC
IR ← MDR
```