ISIMA

Assembleur ARM7TDMI
1

ZZ1
Christophe de Vaulx & Alain Tanguy

ISIMA

Introduction

Langage machine

- Chaque type de processeur ne comprend que son propre langage machine.
- Les instructions en langage machine sont des représentations binaires d'opérations, de données, d'adresses, de registres et de modes d'adressage.
- Il est très fastidieux de programmer directement en langage machine.

Langage machine

- L'**assembleur** (programme d'assemblage) traduit du langage d'assemblage symbolique (**assembleur**) en langage machine.

- En assembleur les opérations, les données, les adresses et les registres sont nommés. La base 10 est utilisable.

- Le premier langage informatique fut l'assembleur introduit en 1947 par Maurice V. Wilkes.

Langage assembleur

- En langage assembleur une instruction machine est représentée sous une forme plus intelligible.

- Le langage assembleur dépend donc fortement du type de processeur utilisé.

⇒Il n'existe pas un langage assembleur, mais un langage assembleur par type de processeur.

⇒Cross-assembler (assembleur générique) difficile à maitriser.

Avantage / Inconvénients

+ Les programmes écrits en assembleur sont très rapides et compacts car très optimisés !!!

- Nécessité de connaître le fonctionnement du processeur et de l'architecture externe (E/S).
- Codes difficiles à comprendre et à déboguer.
- Codes non portables.

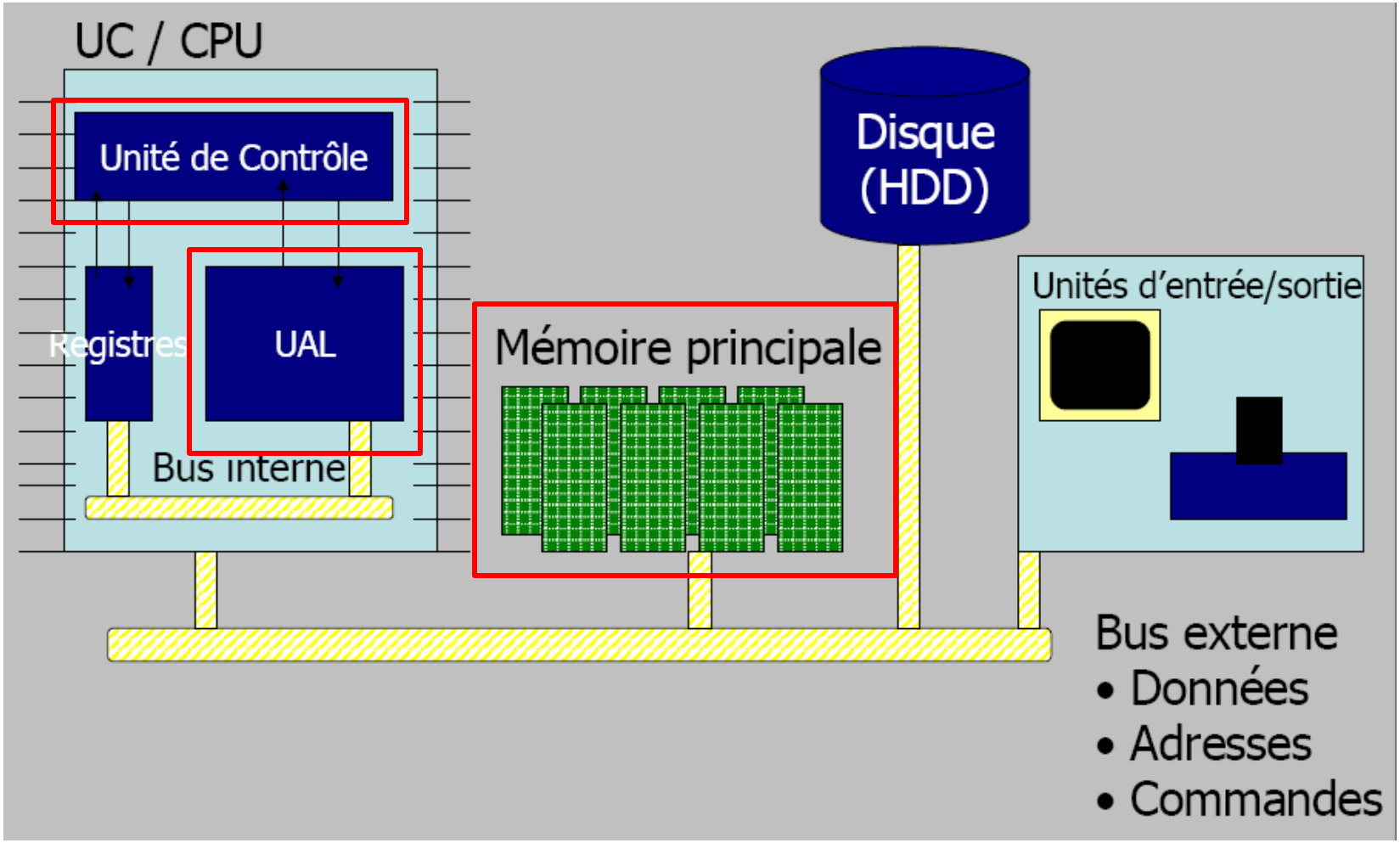
Usages

- Drivers des périphériques.
- Certaines parties des systèmes d'exploitation.
- Extension des fonctionnalités de langages de programmation.
- Systèmes embarqués disposant de peu de mémoire.

ISIMA

Fonctionnement d'un microprocesseur

Schéma fonctionnel d'un ordinateur



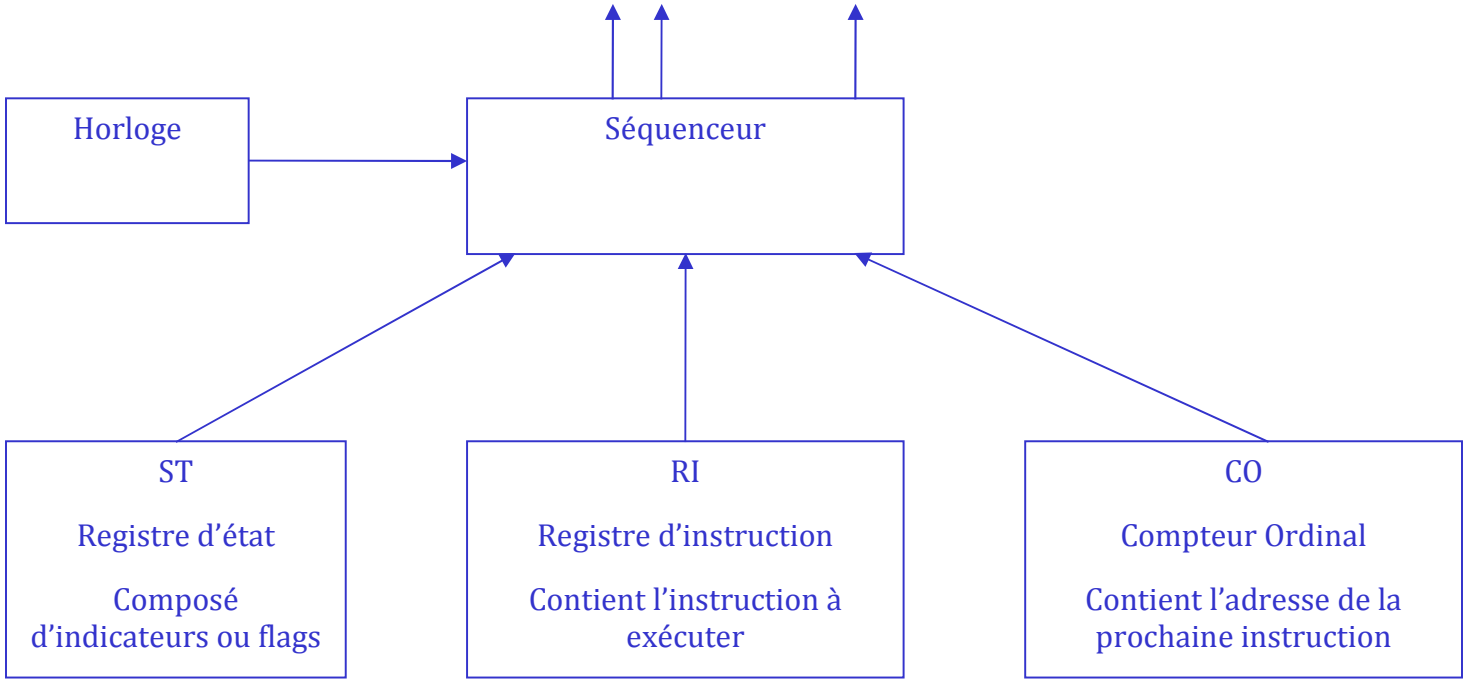
Unité de contrôle ou de commande

L'unité de contrôle est le maître d'œuvre des opérations de transfert de données. Elle sert à

- Adresser l'instruction suivante en mémoire ;
- Décoder l'instruction ;
- Rechercher les données en mémoire ou dans les registres ;
- Envoyer des données dans l'unité de traitement ;
- Indiquer l'opération que l'unité de traitement doit effectuer ;
- Transférer éventuellement le résultat.

Unité de contrôle ou de commande

Structure de l'unité de contrôle



Unité de contrôle ou de commande

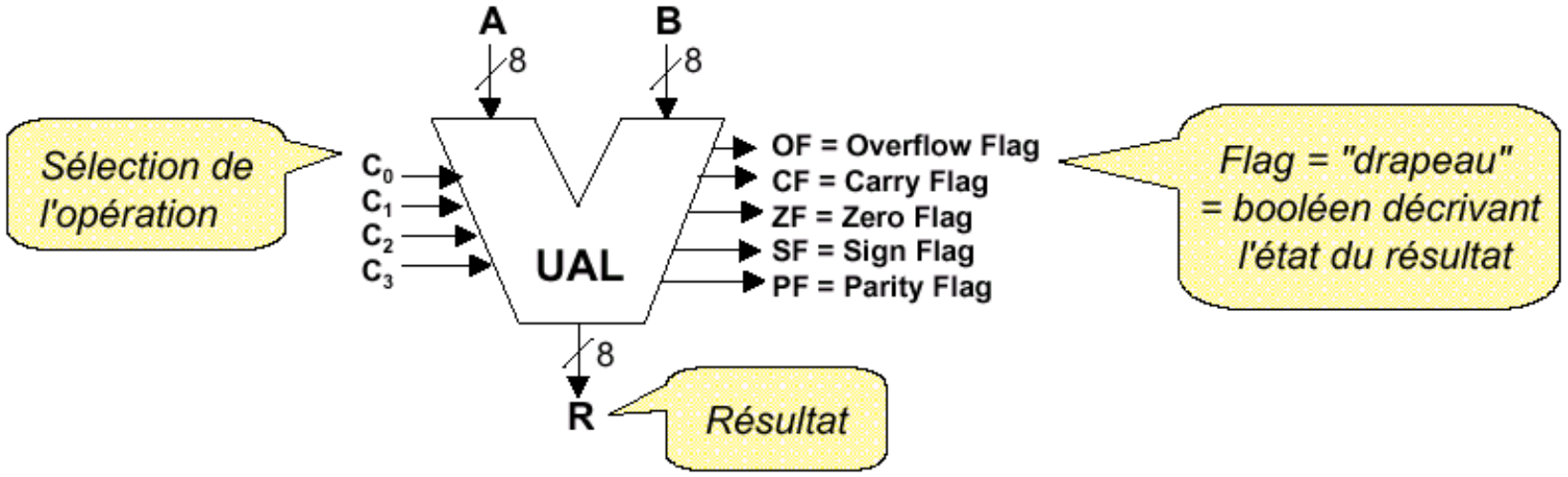
Elle s'appuie le plus souvent sur des séquenceurs locaux et spécialisés

- Coprocesseurs arithmétiques ;
- Arbitres de bus ;
- Contrôleur d'interruption ;
- Contrôleurs de périphériques de l'unité d'échange.

Unité arithmétique et logique (UAL) ou de traitement

Elle est chargée d'effectuer :

- Les opérations arithmétiques : +, -, *, /, +1, -1…
- Les opérations logiques : et, ou, non, décalages...



Mémoires

- La mémoire est la partie de l'ordinateur dans laquelle les programmes et les données sont stockés.
- Il existe plusieurs types de mémoires :
  - Mémoire auxiliaire (grosse capacité, lente et peu onéreuse) : disque dur, clés USB ;

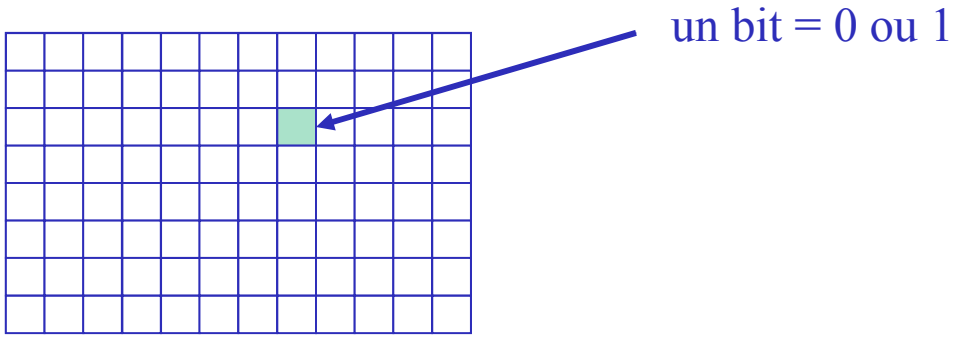
Mémoires

– Mémoire centrale (moyenne, rapide et onéreuse) : RAM et ROM ;

– Mémoire locale (petite, très rapide et très onéreuse) : mémoire cache.

Mémoires

- Ces mémoire peuvent être :
  - Volatiles : RAM, cache ;
  - Non volatiles : ROM, DD, mémoire flash.
- De manière imagée, on peut se représenter la mémoire d'un ordinateur comme un ensemble de petites cases appelées **bits**.



Mémoires

- Pour qu'un programme puisse retrouver les données et les instructions stockées dans la mémoire cette dernière est adressée.
- Pour adresser une mémoire, on la découpe d'abord en cellules formées de plusieurs bits.
- Toutes les cellules d'une mémoire comportent le même nombre de bits.

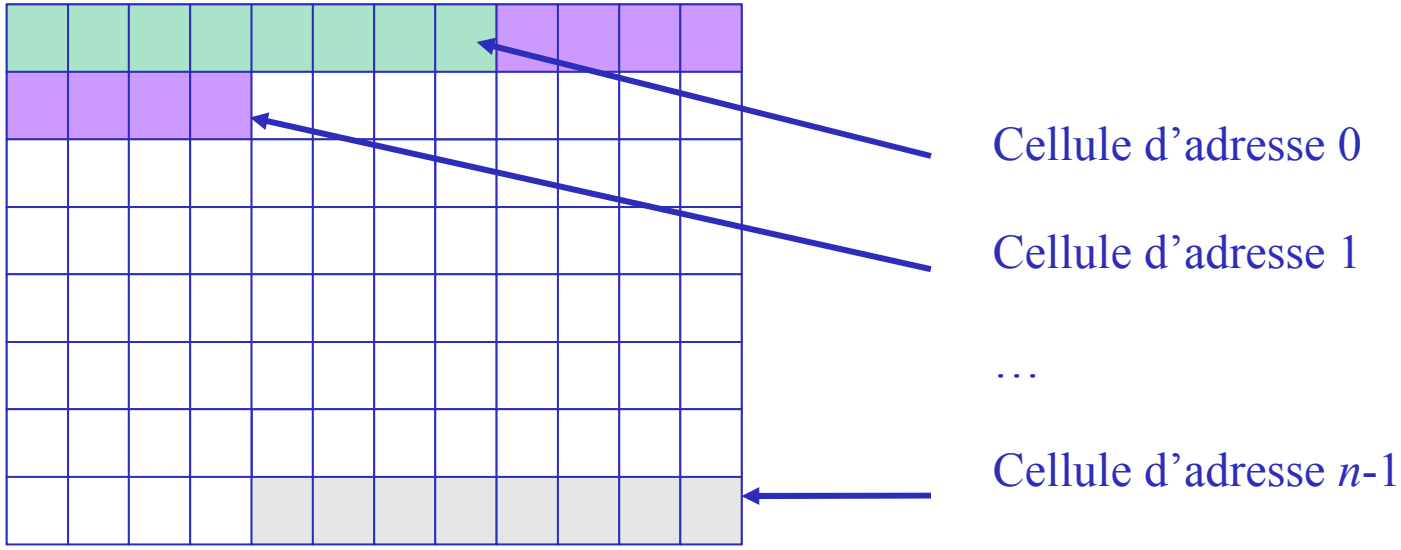
Mémoires

- On attribue un numéro à chacune des cellules.
- Ce numéro s'appelle adresse il permet de référencer la cellule.

Remarque

Si une mémoire est découpée en *n* cellules, la première cellule aura l'adresse 0 et la dernière l'adresse *n*-1.

Mémoires



La taille de cellules adressées varie d'un processeur à l'autre. Les tailles de cellules les plus utilisées sont 8, 16 et 32 bits.

Mémoires

Remarques

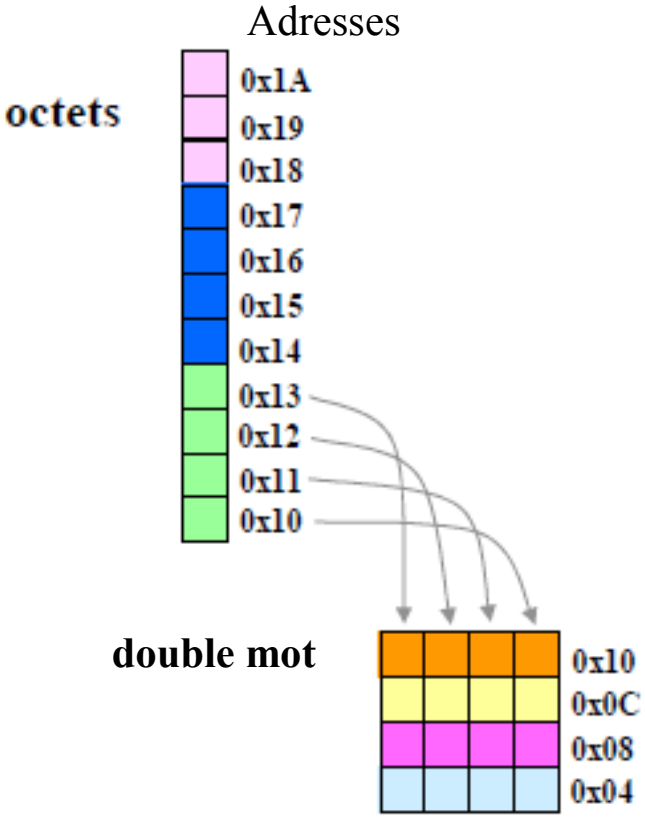
- Une cellule de 8 bits est un octet (byte)
- Une cellule de 16 bits est un mot (word)
- Une cellule de 32 bits est un double mot (double word)
- Pour l'ARM : octet, demi mot (half word 16), mot (32)

Une suite de 1024 octets est un Ko (Kilo-octet)
1 Mo correspond à 1024 Ko
1 Go à 1024 Mo, puis To, Po...

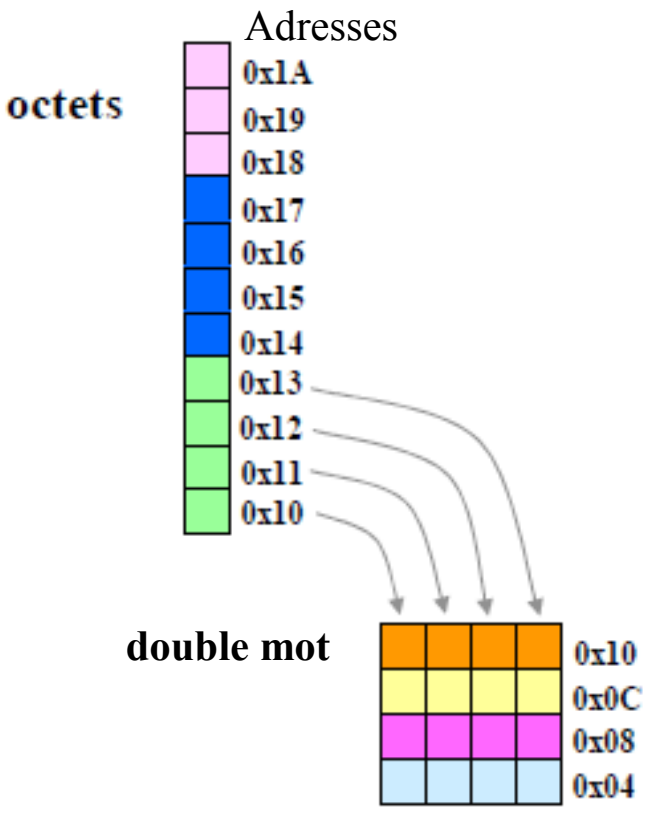
Mémoires

- Au sein d'un mot et d'un double mot, il y 2 façons d'organiser les octets.
- Ces organisation sont appelées (Gulliver) :
  « Little Endian » poids faibles d'abord et
  « Big Endian » poids forts d'abord .
- Les premiers microprocesseurs 16 bits Intel 8086 et Motorola 68000 avaient un type différent.

Mémoires



« Little Endian »



« Big Endian »

Échanges entre le processeur et la mémoire

- Le processeur exécute un programme en mémoire

=>Transfert d'**instructions**

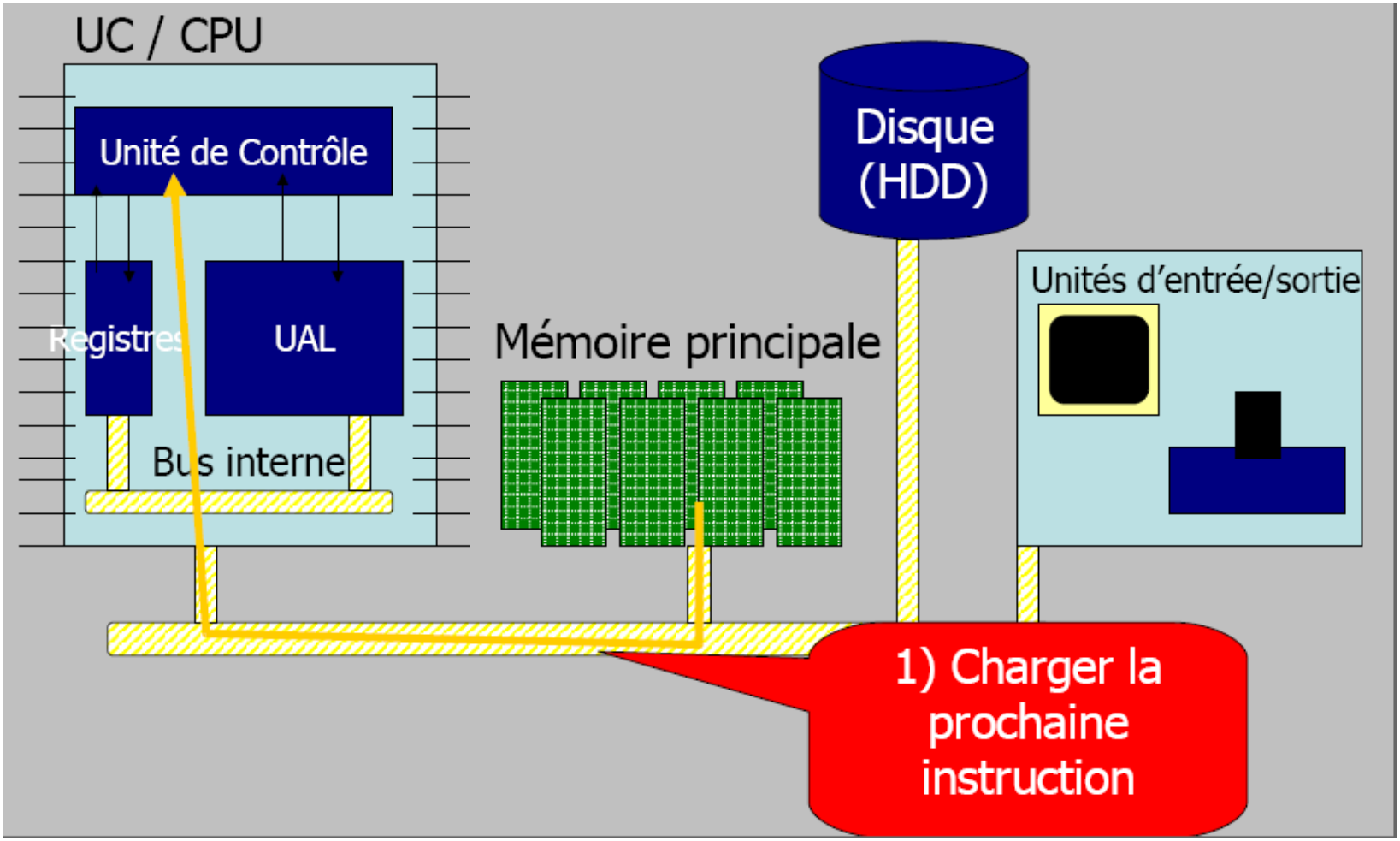
- Le programme manipule des variables

=>Transfert de **données**

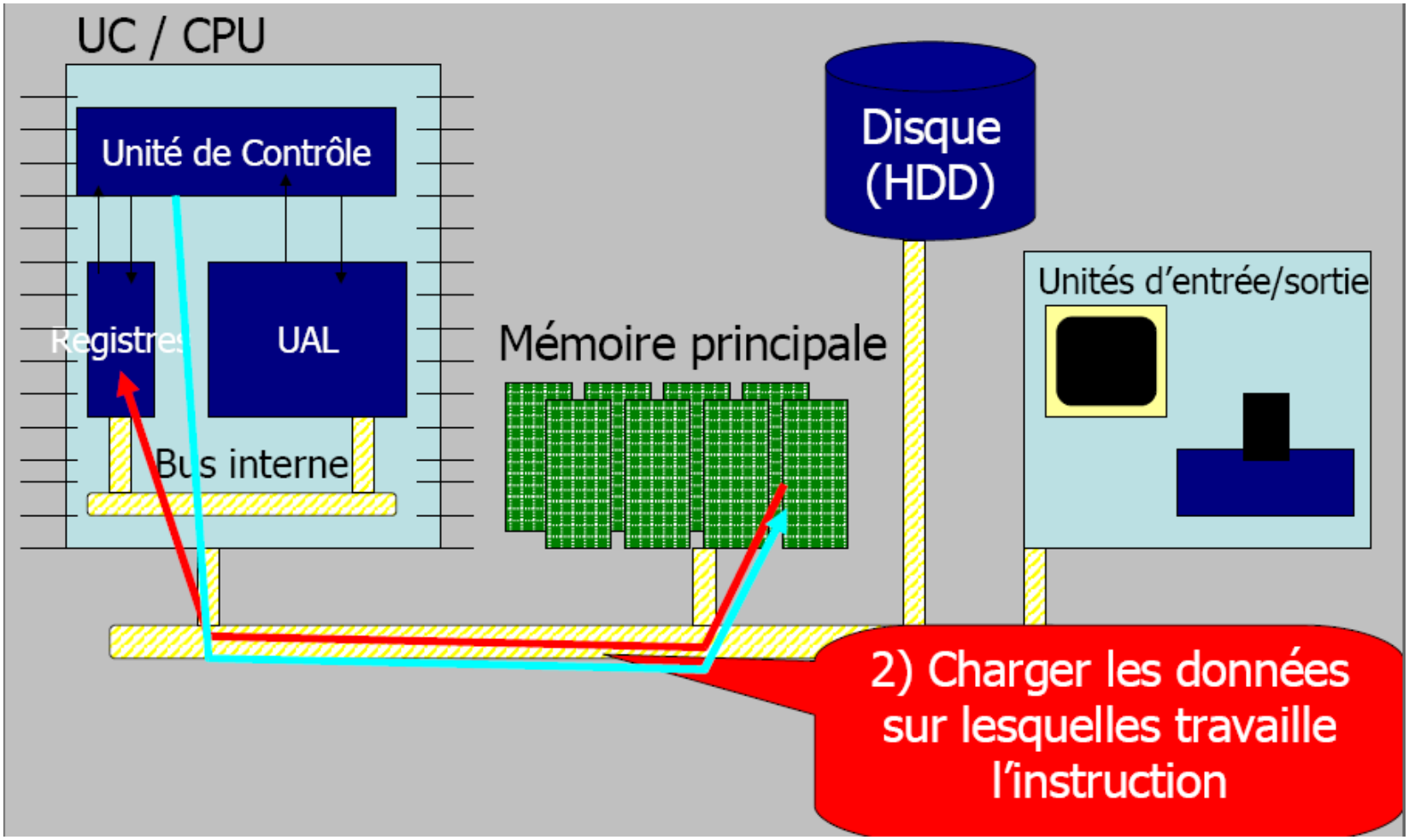
- Ces informations sont rangées à un certain emplacement en mémoire

=>Transfert d'**adresses**

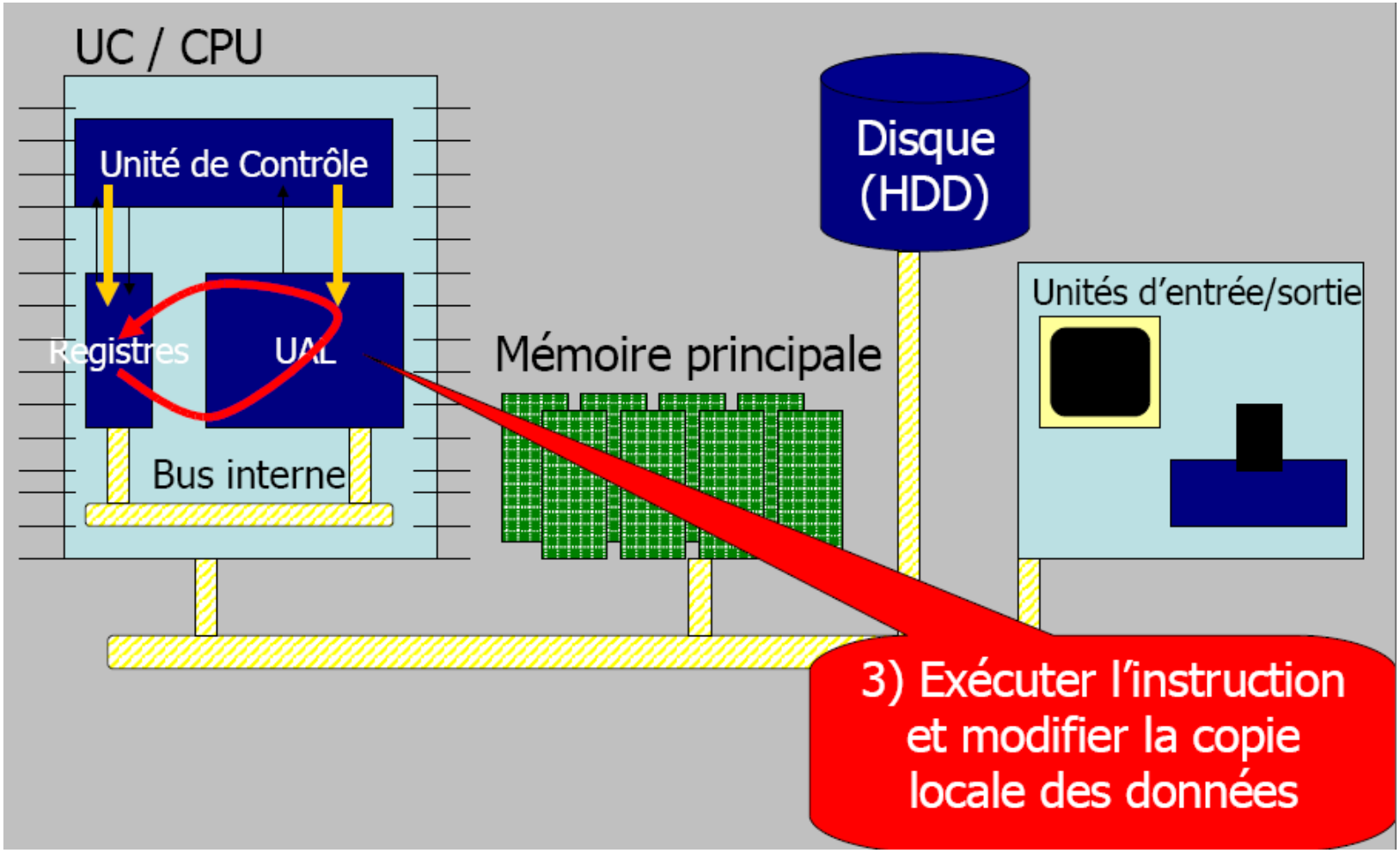
Principe du déroulement de l'exécution d'une instruction



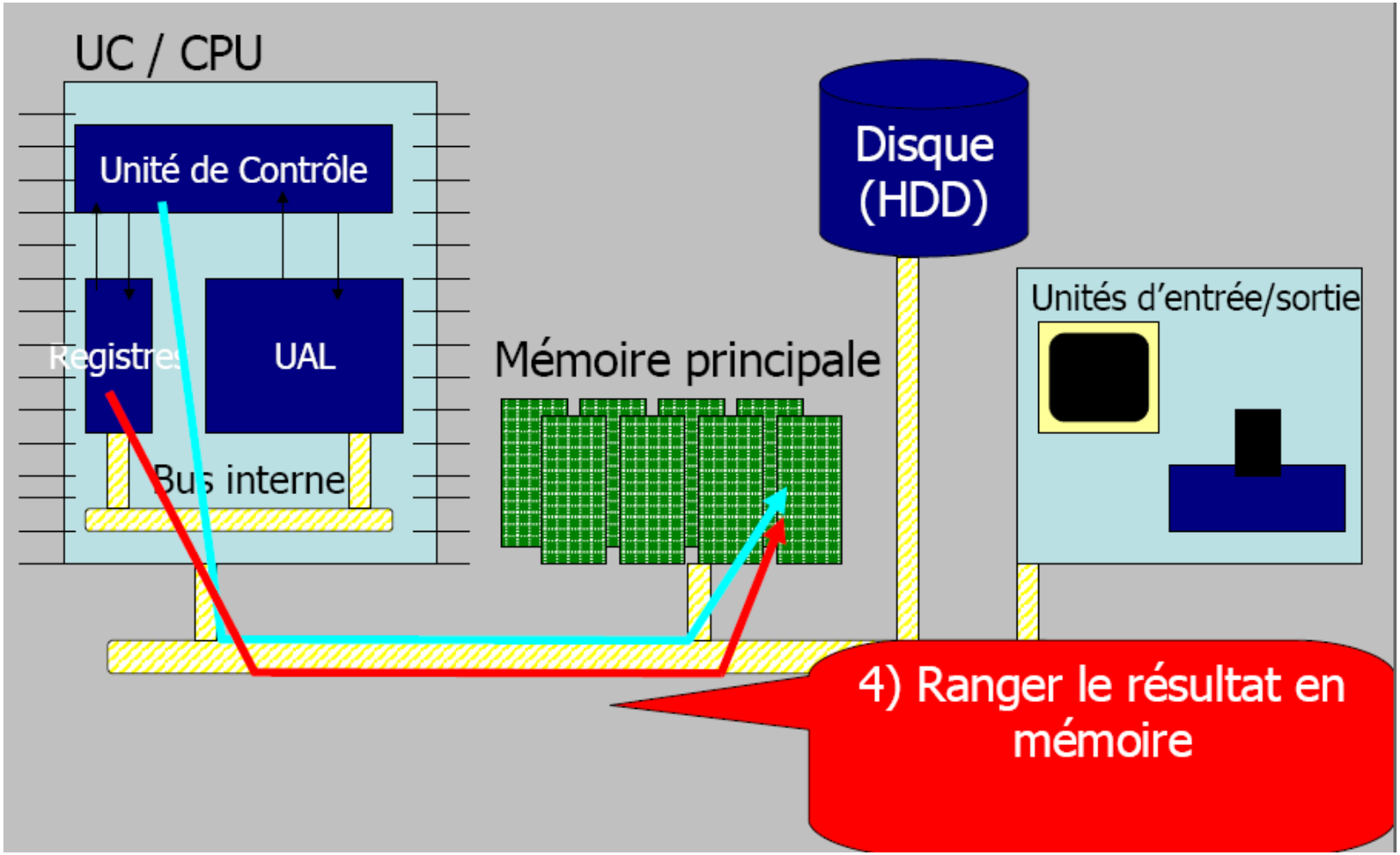
Principe du déroulement de l'exécution d'une instruction



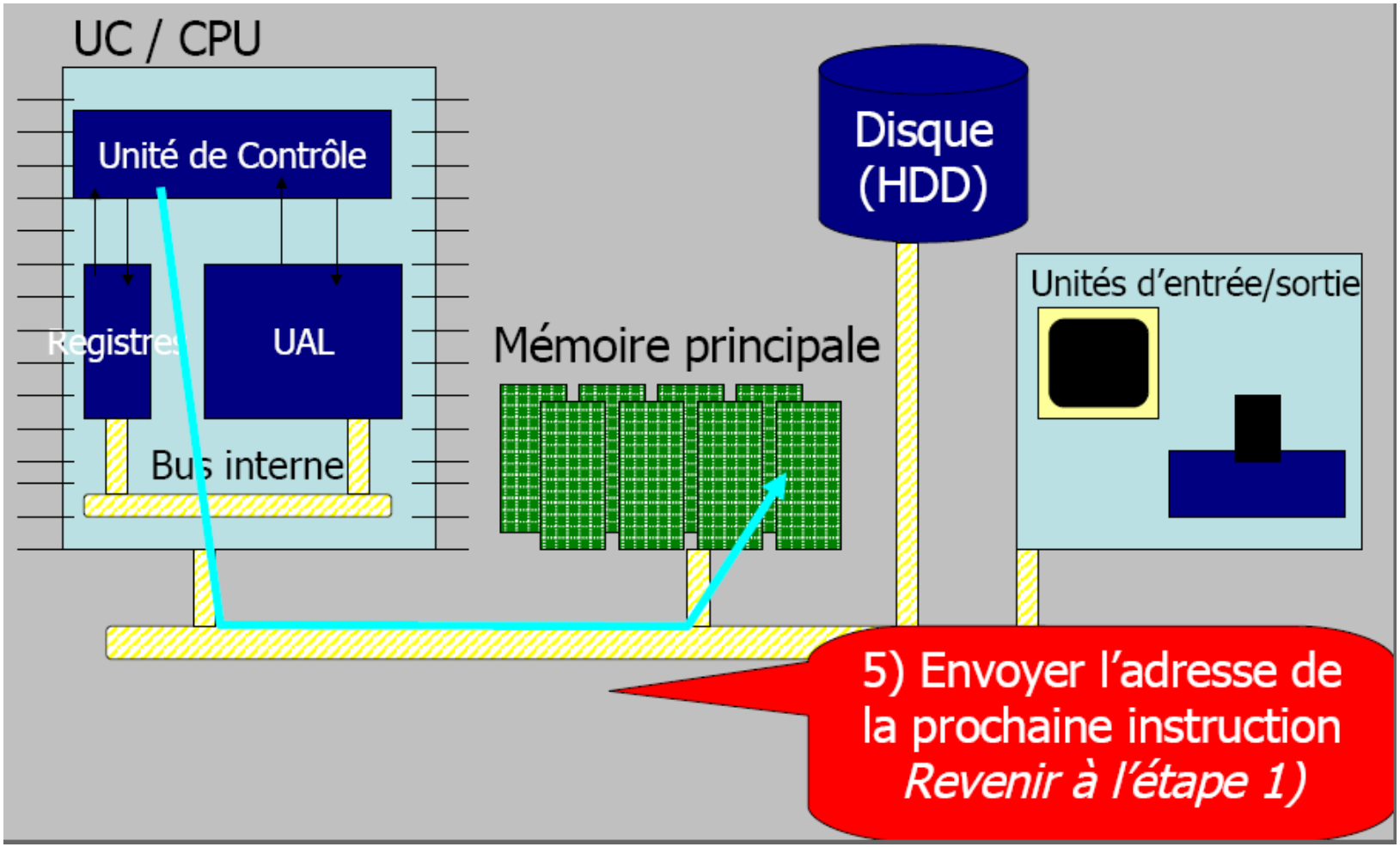
Principe du déroulement de l'exécution d'une instruction



Principe du déroulement de l'exécution d'une instruction



Principe du déroulement de l'exécution d'une instruction



Exécution d'une instruction

Le processeur exécute une instruction en 8 étapes

cycle de chargement-décodage-exécution

1. Chargement de la prochaine instruction à exécuter depuis la mémoire dans le registre instruction,
2. Modification du compteur ordinal pour qu'il pointe sur l'instruction suivante,
3. Décodage de l'instruction que l'on vient de charger,
4. Localiser dans la mémoire les données utilisées par l'instruction,
5. Chargement des données si nécessaire de la mémoire dans les registre internes du micro processeur,
6. Exécution de l'instruction,
7. Stockage des résultats à leurs destinations respectives,
8. Retour à l'étape 1 pour exécuter l'instruction suivante.

Jeu d'instructions

- Un processeur peut exécuter un certain nombre d'instructions très simples.
- L'ensemble de ces instructions est appelé jeu d'instructions du processeur.
- On distingue deux types processeurs :
  - les processeurs CISC ont un jeu d'instructions complexe ce qui permet de réduire la taille du code,
  - les processeurs RISC ont un jeu d'instructions réduit mais dont la vitesse d'exécution est optimisée...

Jeu d'instructions

Jeu d'instructions d'un processeur

- Traitements Arithmétiques et Logiques
- Transfert de données
- Entrée/Sortie
- Ruptures de séquence
  - Branchements inconditionnels et conditionnels
  - Appel de sous-programme, Retour et Gestion d'interruption
- Contrôle externe
  - Verrouillage de bus
  - Interruption matérielle

Format d'une instruction

- Le format général d'une instruction est
  CodeOpération $Opérande_1$, ..., $Opérande_n$
- Le code de l'opération permet au processeur de savoir quelle opération effectuer.
- Les opérandes sont des registres, des adresses, des identificateurs ou des constantes.
- Les modes d'adressage sont représentées à l'aide de symboles [...] # @ ...

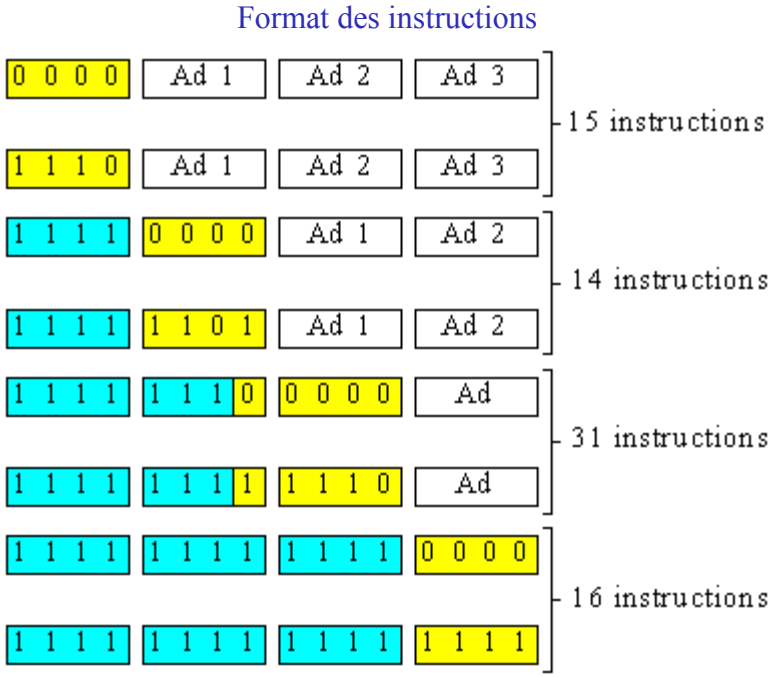
Format d'une instruction

Exemples d'instructions assembleur 8086

- ADD AX, 2
- MOV AX, [BX]
- INC BX
- NOP

Le nombre d'octets nécessaires pour coder une instruction varie selon les processeurs et selon les instructions.

Format d'une instruction



PDP 11 :
Instructions de tailles fixes
CodeOp expansif

Format d'une instruction

OpCode	R1	R2
8	4	4

OpCode	R1	X2	B2	D2
8	4	4	4	12

OpCode	R1	R3	B2	D2
8	4	4	4	12

OpCode	Const	C1	D1
8	8	4	12

OpCode	Length	B1	D1	B2	D2
8	8	4	12	4	12

IBM 370 : Instructions de tailles variables

Modes d’adressage

- Une donnée manipulée par une instruction peut être adressées de différentes façons :

  l'adresse peut être rangée dans un registre, dans une variable en mémoire ou directement dans l'instruction.

- On appelle mode d’adressage, la façon d'accéder à une donnée soit par calcul d'adresse mémoire effective, soit par sélection d'un registre ou par extraction à partir du code de l'instruction.

Exemples de modes d'adressage

Adressage immédiat : la donnée est codée dans l'instruction.

Adressage registre direct : l'instruction indique dans quel registre se trouve la donnée.

Adressage mémoire direct : l'instruction indique à quelle adresse mémoire se trouve la donnée.

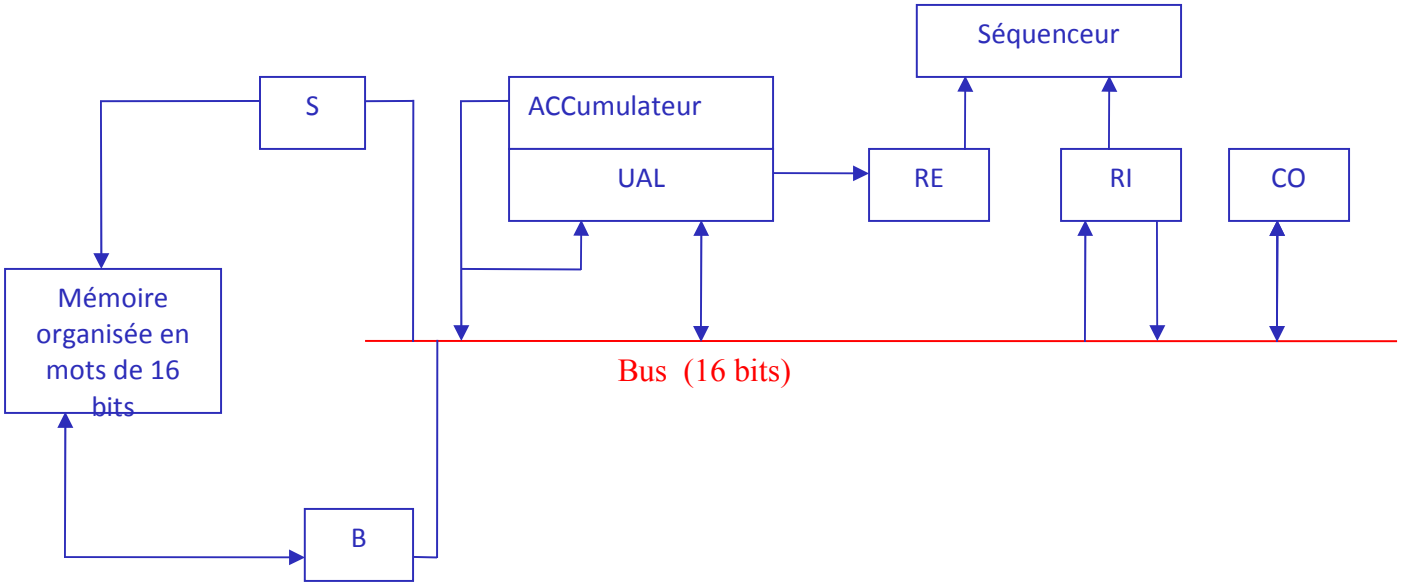
Adressage registre indirect : l'instruction indique dans quel registre se trouve l'adresse de la donnée.

Adressage indirect : l'instruction contient l'adresse de l'adresse de la donnée (pointeur).

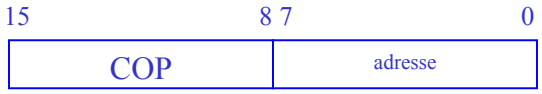
Notation

Dans la suite de ce cours le contenu du mot d'adresse a sera noté (a) !!!

Exemple d'un processeur 16 bits à 1 bus

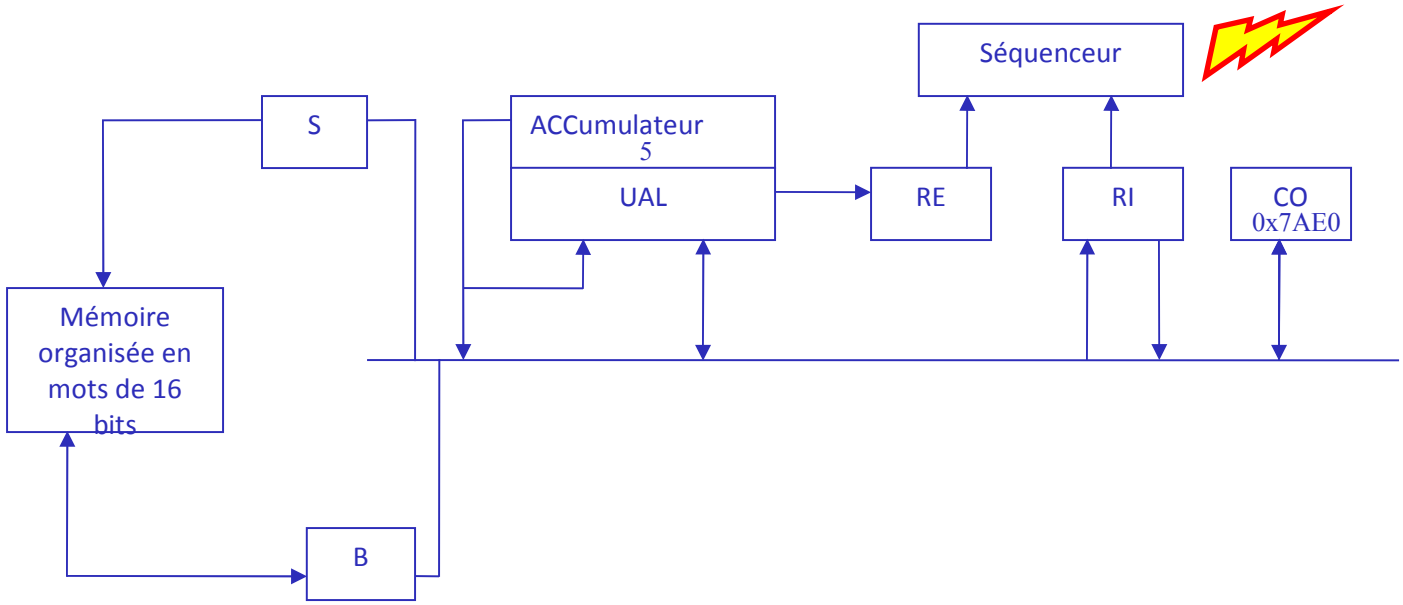


Format des instructions :



Exemple d'un processeur 16 bits à 1 bus

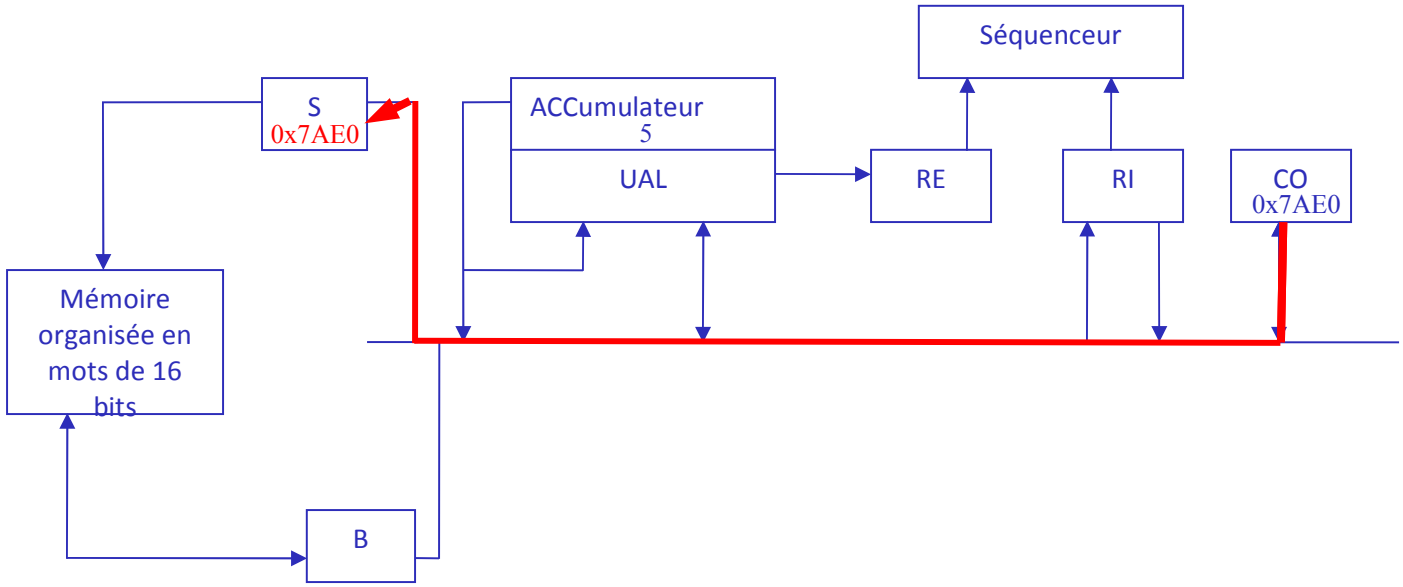
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



1- Le séquenceur délivre un signal pour initialiser le cycle

Exemple d'un processeur 16 bits à 1 bus

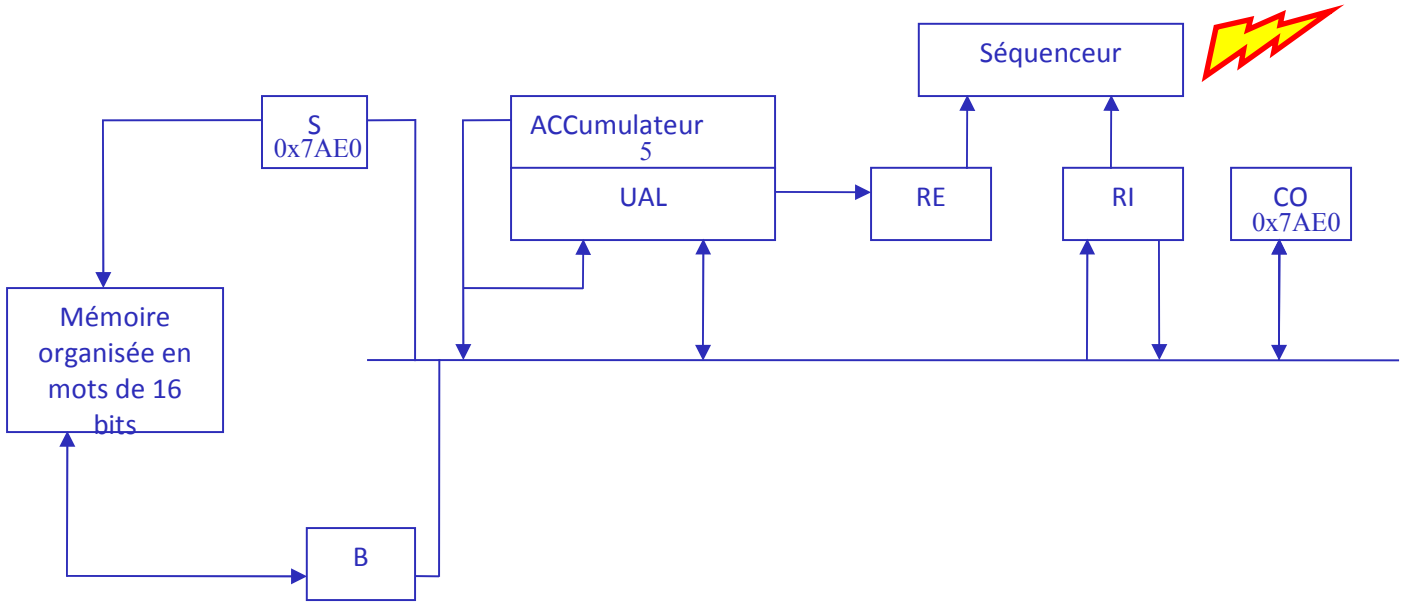
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



2- Le contenu du compteur ordinal est placé dans le registre d'adresses de la mémoire via le bus : (S) := (CO)

Exemple d'un processeur 16 bits à 1 bus

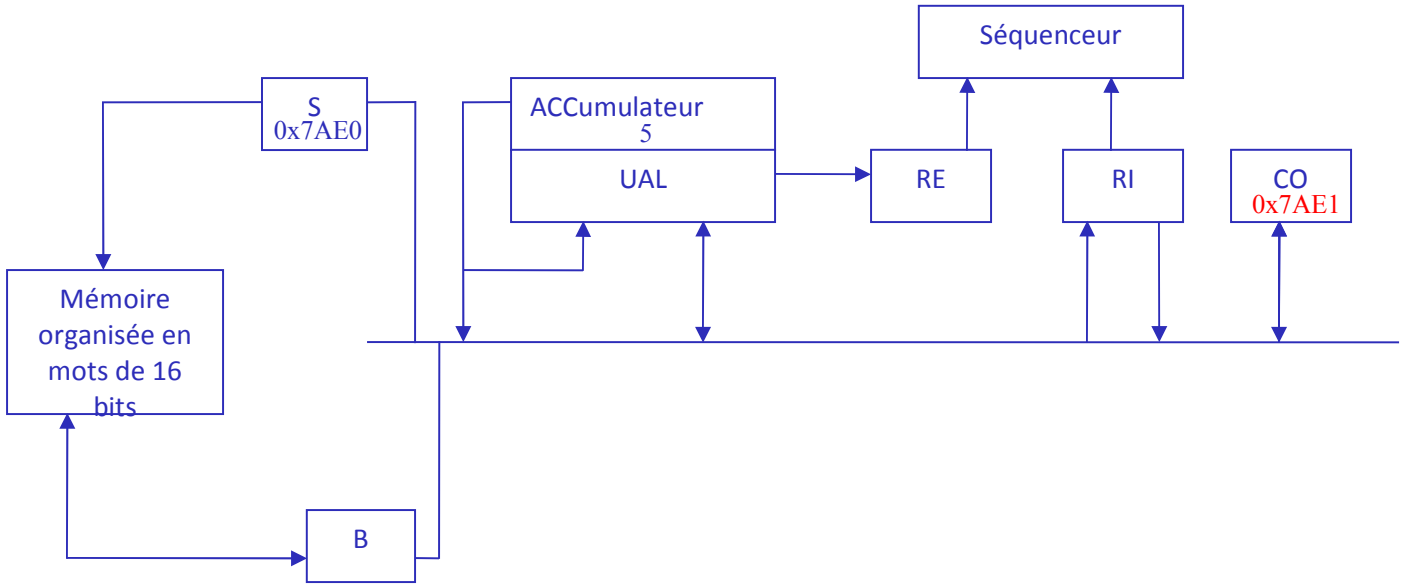
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



3- Le séquenceur délivre un signal au compteur ordinal

Exemple d'un processeur 16 bits à 1 bus

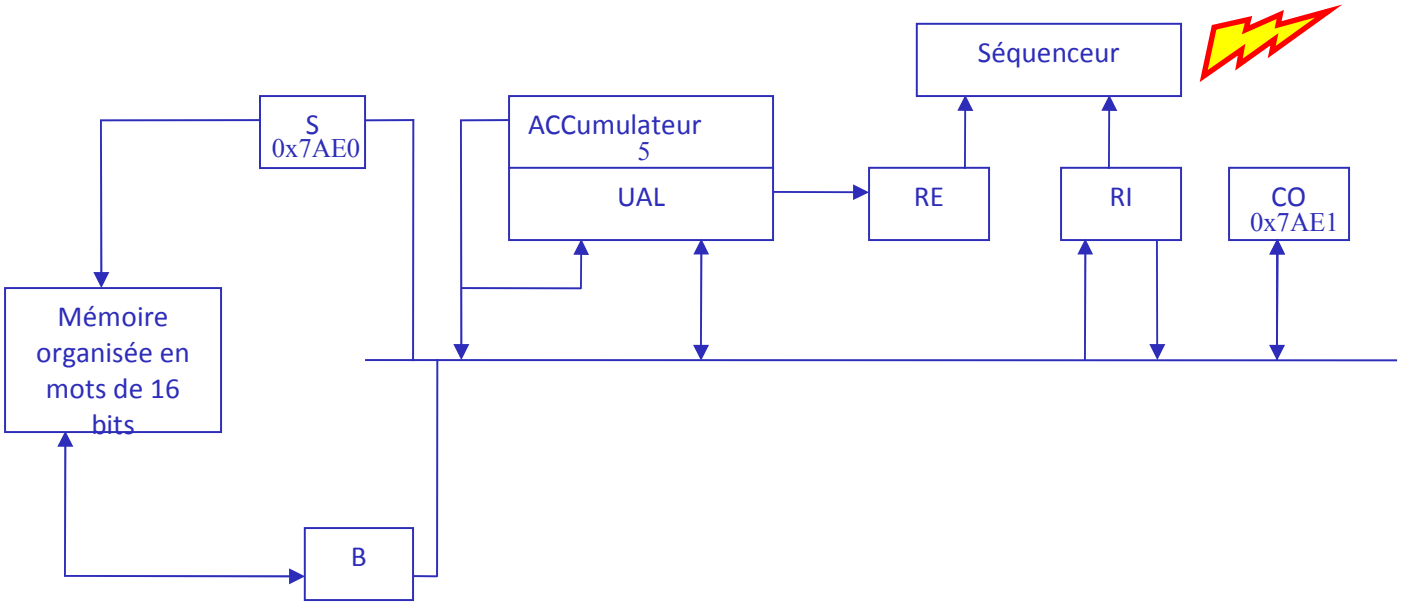
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



4- Le contenu du compteur ordinal est incrémenté :
(CO) := (CO)+1

Exemple d'un processeur 16 bits à 1 bus

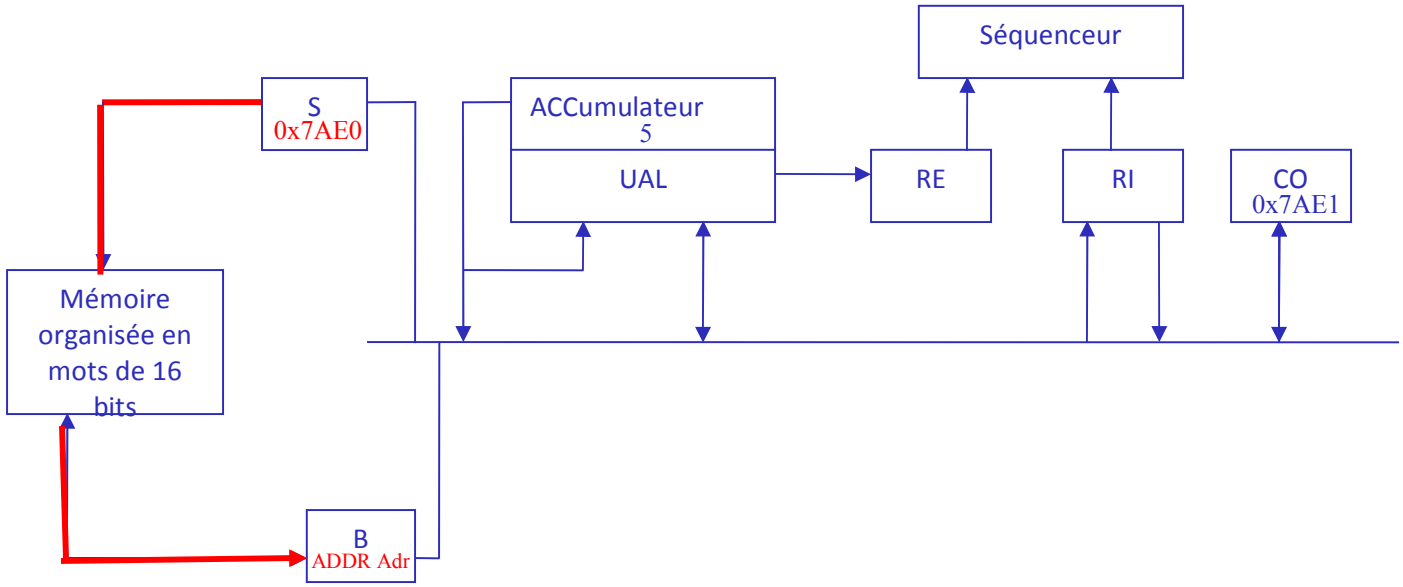
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



5- Le séquenceur délivre un signal de lecture au contrôleur de la mémoire

Exemple d'un processeur 16 bits à 1 bus

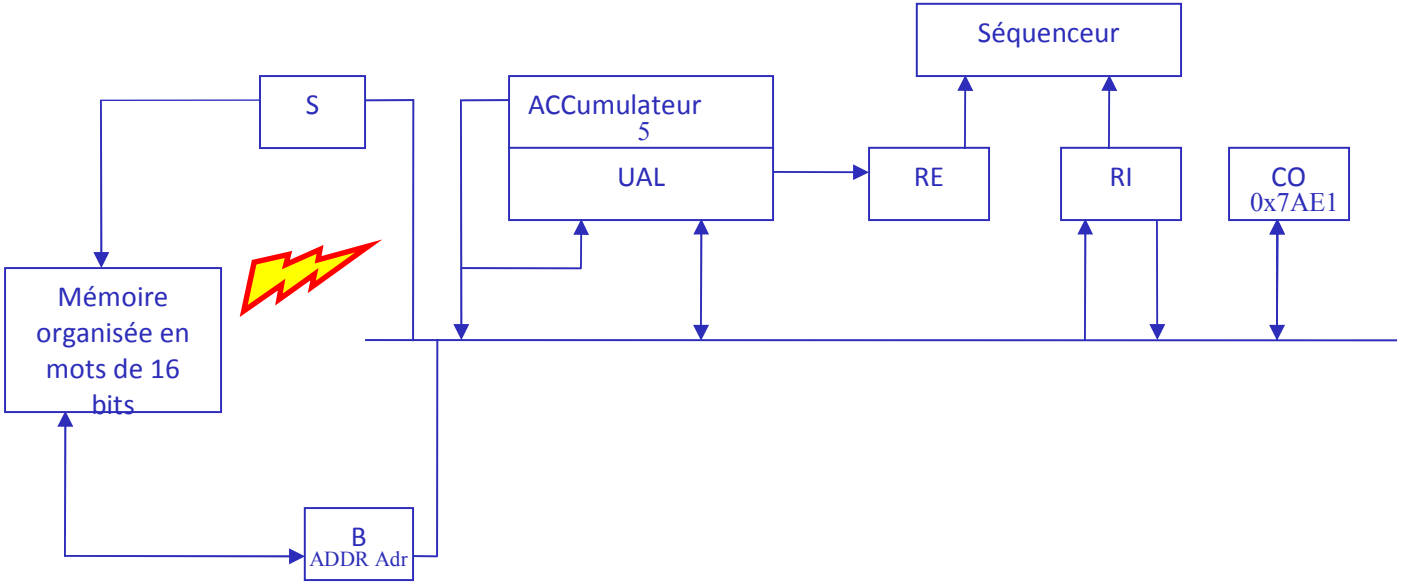
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



6- Le contrôleur de la mémoire place l'instruction dont l'adresse est stockée dans S dans le registre de données B : (B) := ((S))

Exemple d'un processeur 16 bits à 1 bus

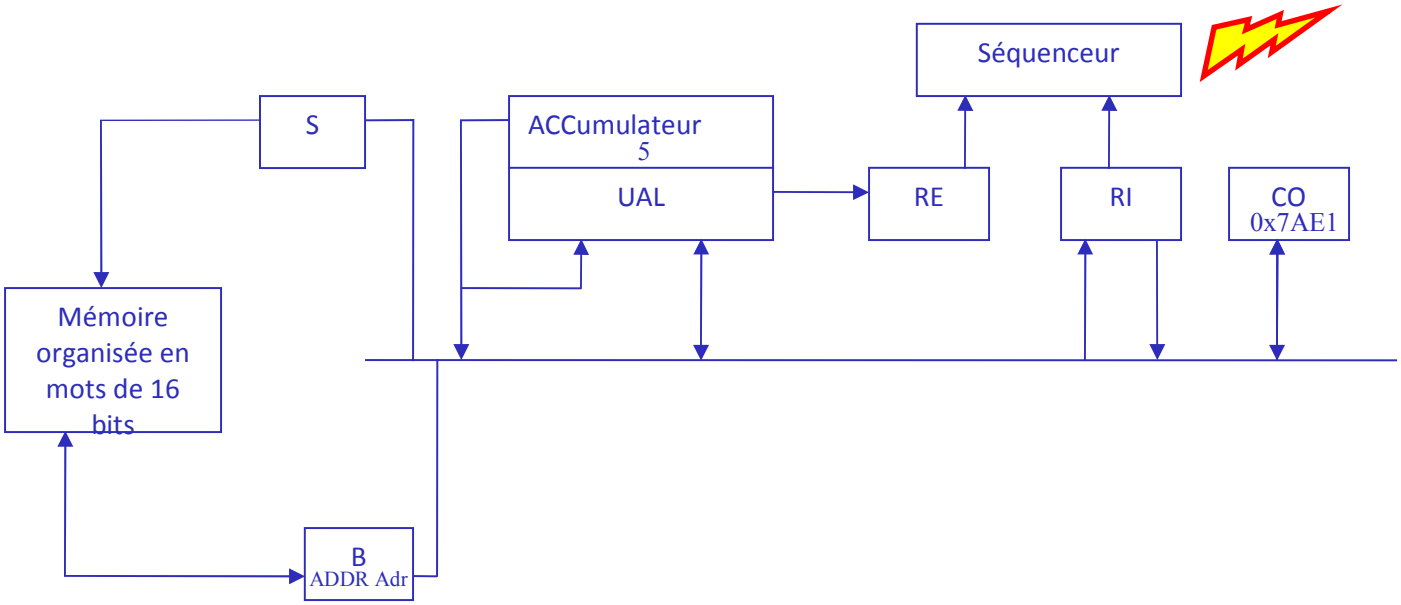
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



7- Le contrôleur de la mémoire délivre un signal de validation

Exemple d'un processeur 16 bits à 1 bus

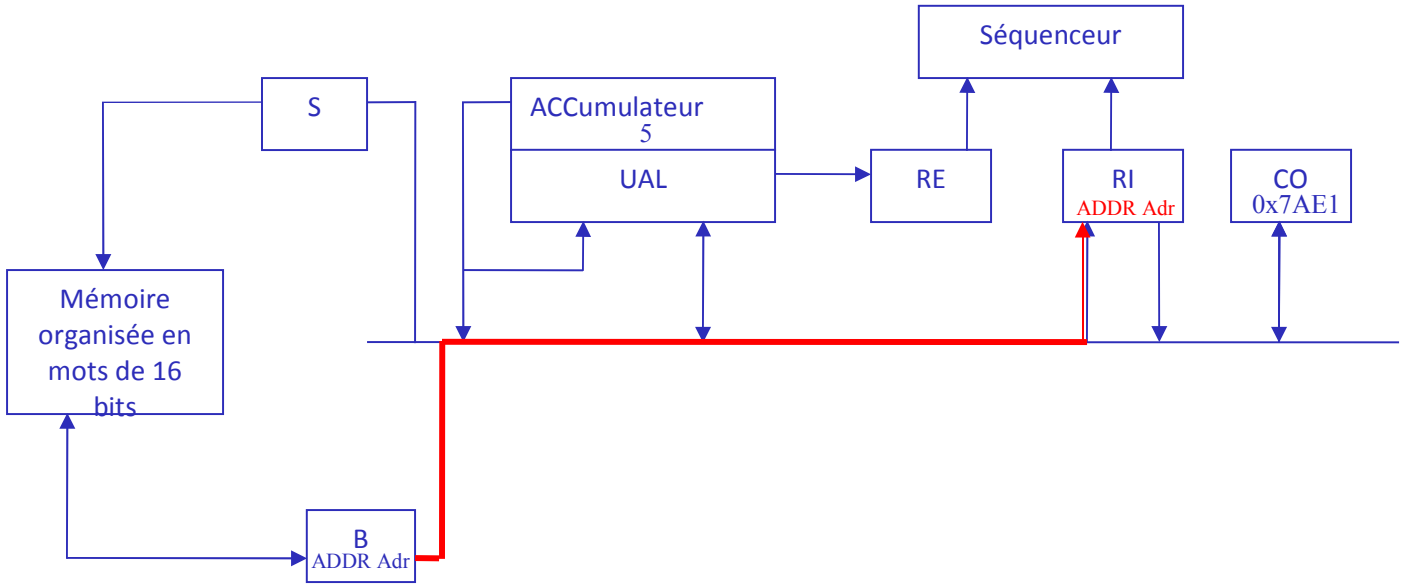
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



8- Le séquenceur délivre un signal de validation

Exemple d'un processeur 16 bits à 1 bus

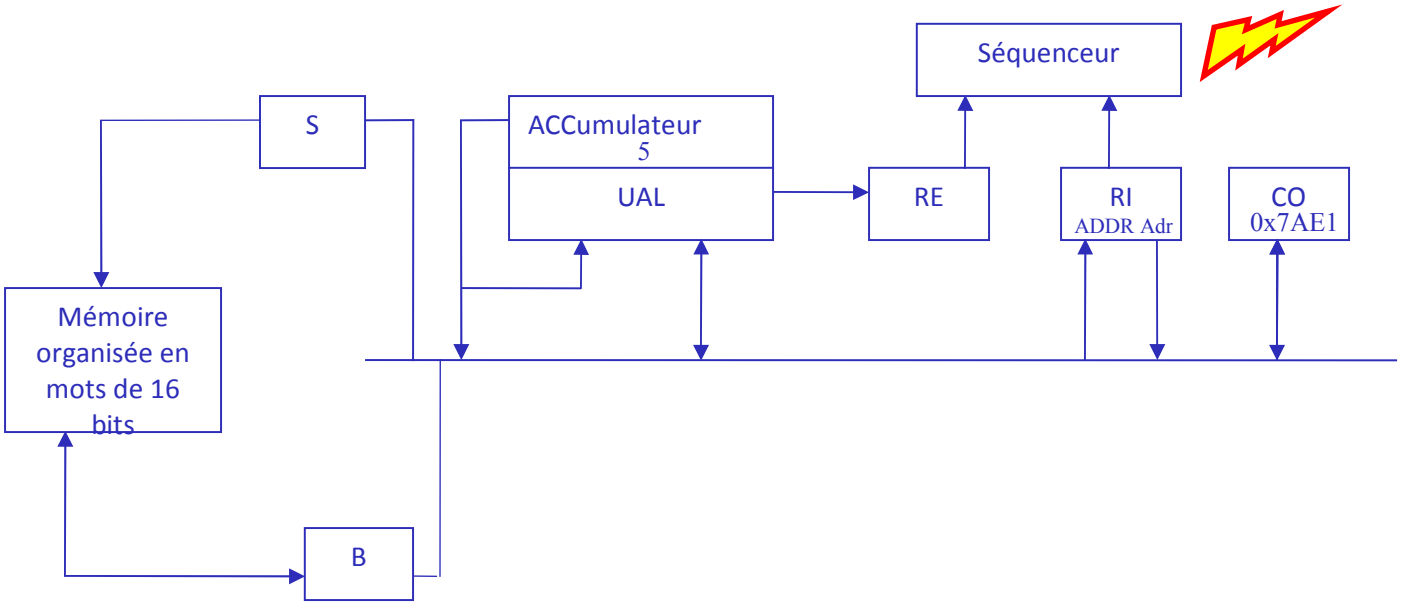
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



9- L'instruction est placée dans le registre d'instructions RI :
(RI) := (B)

Exemple d'un processeur 16 bits à 1 bus

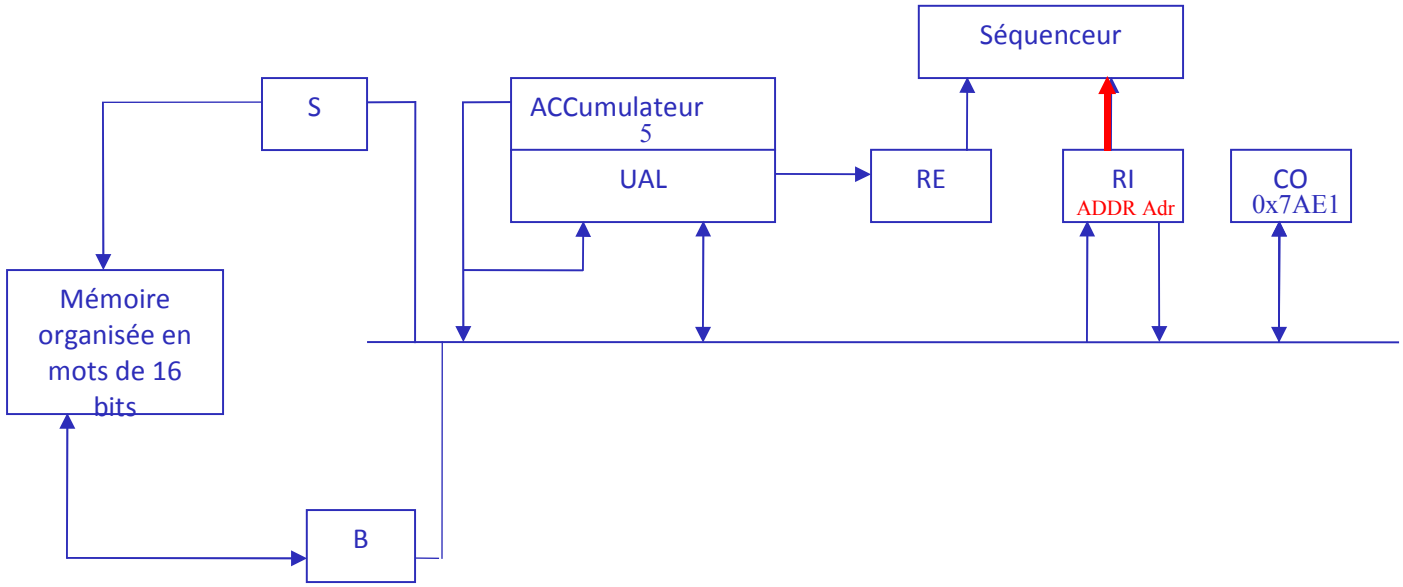
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



10- Le séquenceur délivre un signal de décodage

Exemple d'un processeur 16 bits à 1 bus

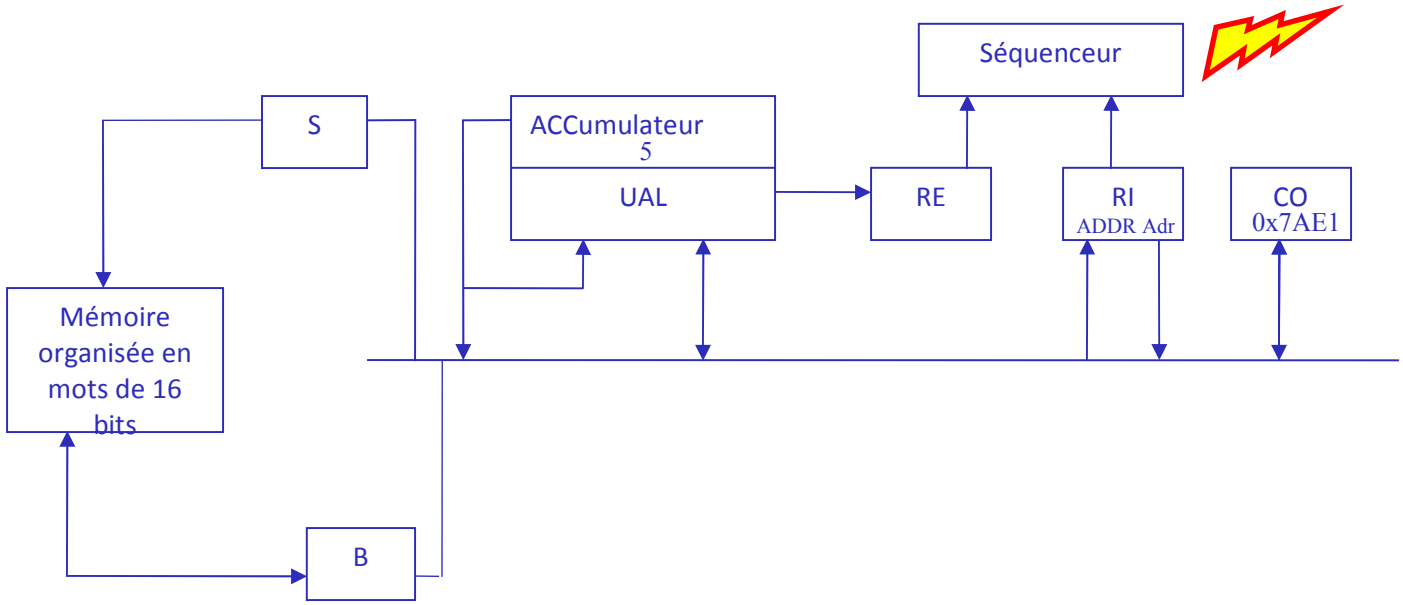
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



11- L'instruction est décodée

Exemple d'un processeur 16 bits à 1 bus

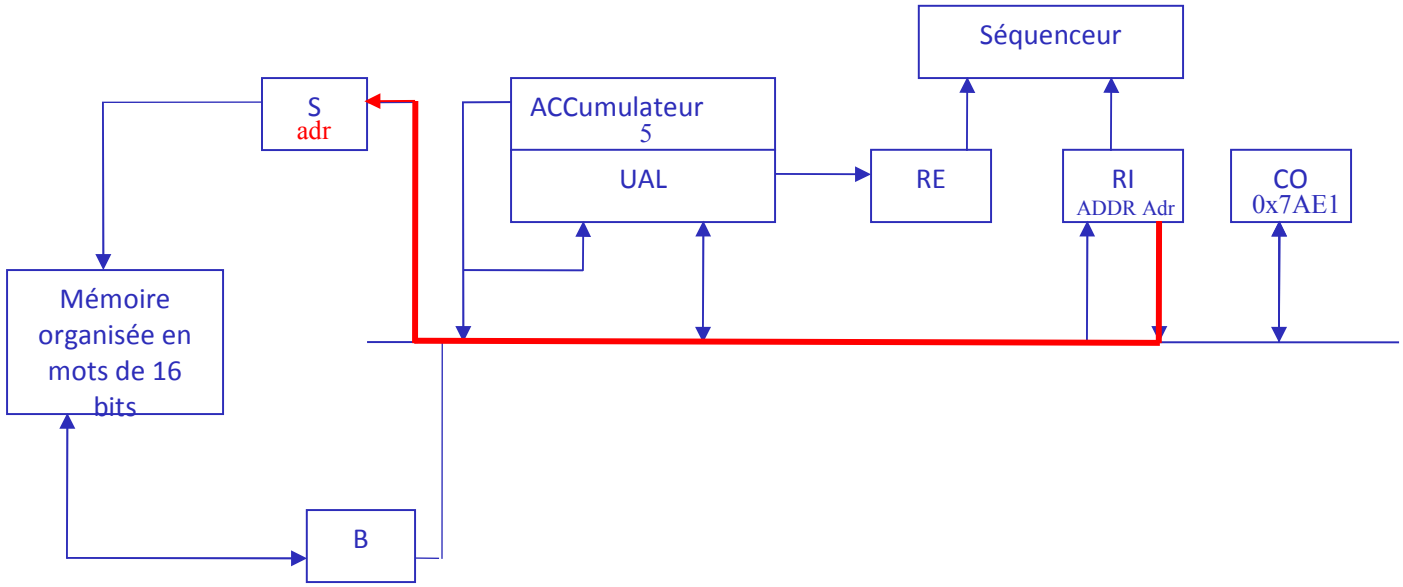
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



12- Le séquenceur délivre un signal

Exemple d'un processeur 16 bits à 1 bus

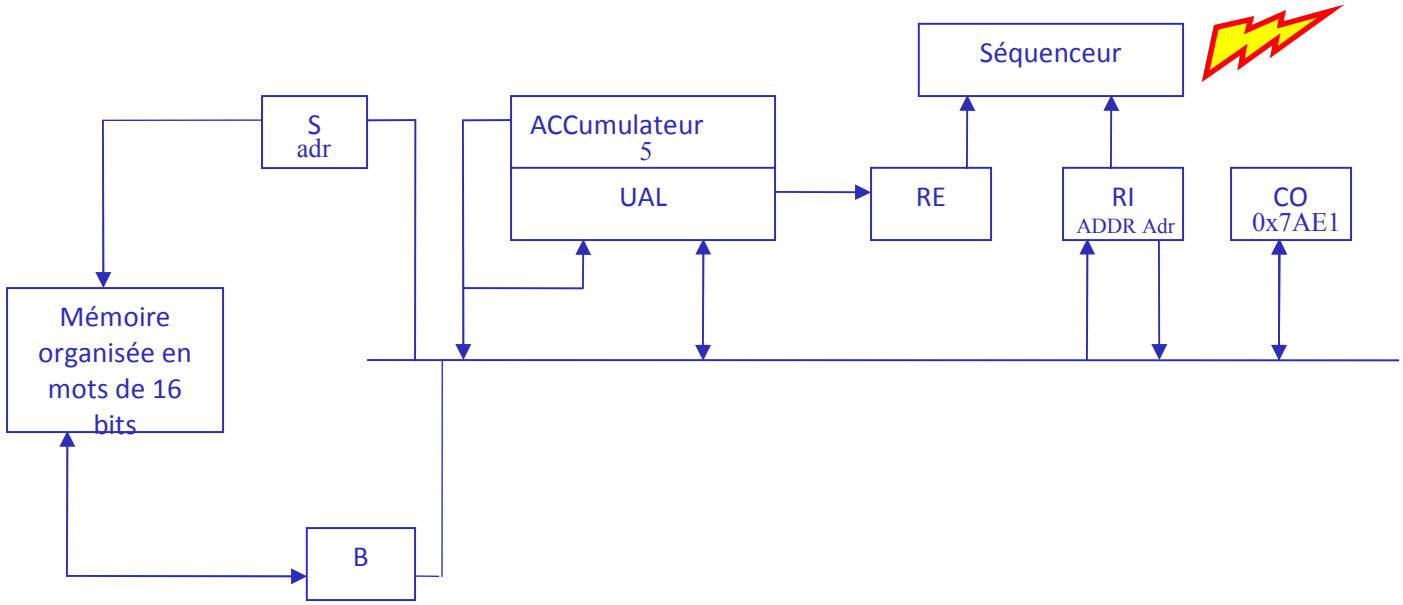
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



13- L'adresse contenu dans l'instruction est placée dans le registre d'adresses de la mémoire : (S) := (adr)

Exemple d'un processeur 16 bits à 1 bus

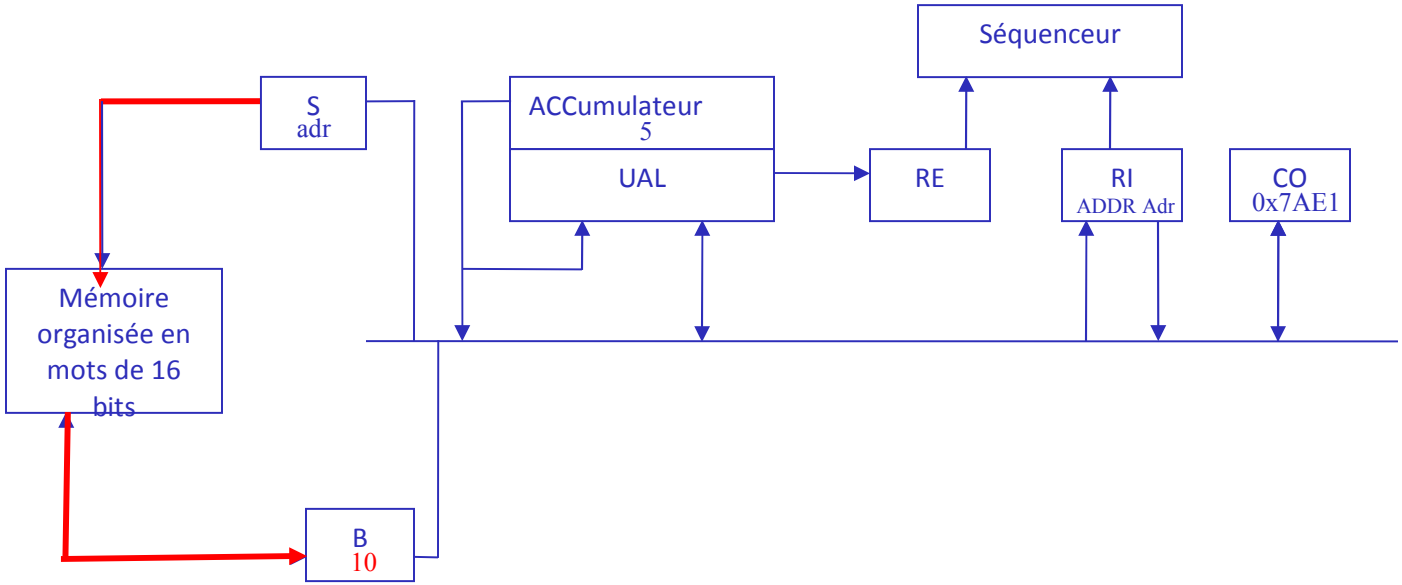
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



14- Le séquenceur délivre un signal de lecture au contrôleur de la mémoire

Exemple d'un processeur 16 bits à 1 bus

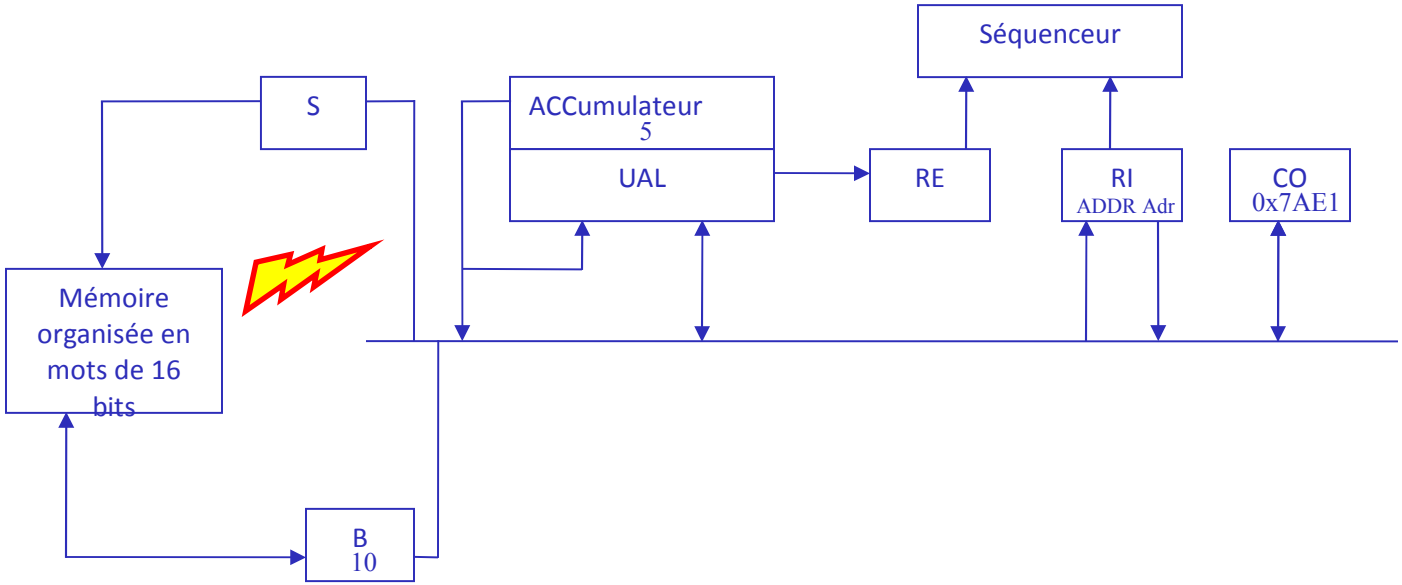
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



15- Le contrôleur de la mémoire place la donnée dont l'adresse est stockée dans S dans le registre de données B : (B) := ((S))

Exemple d'un processeur 16 bits à 1 bus

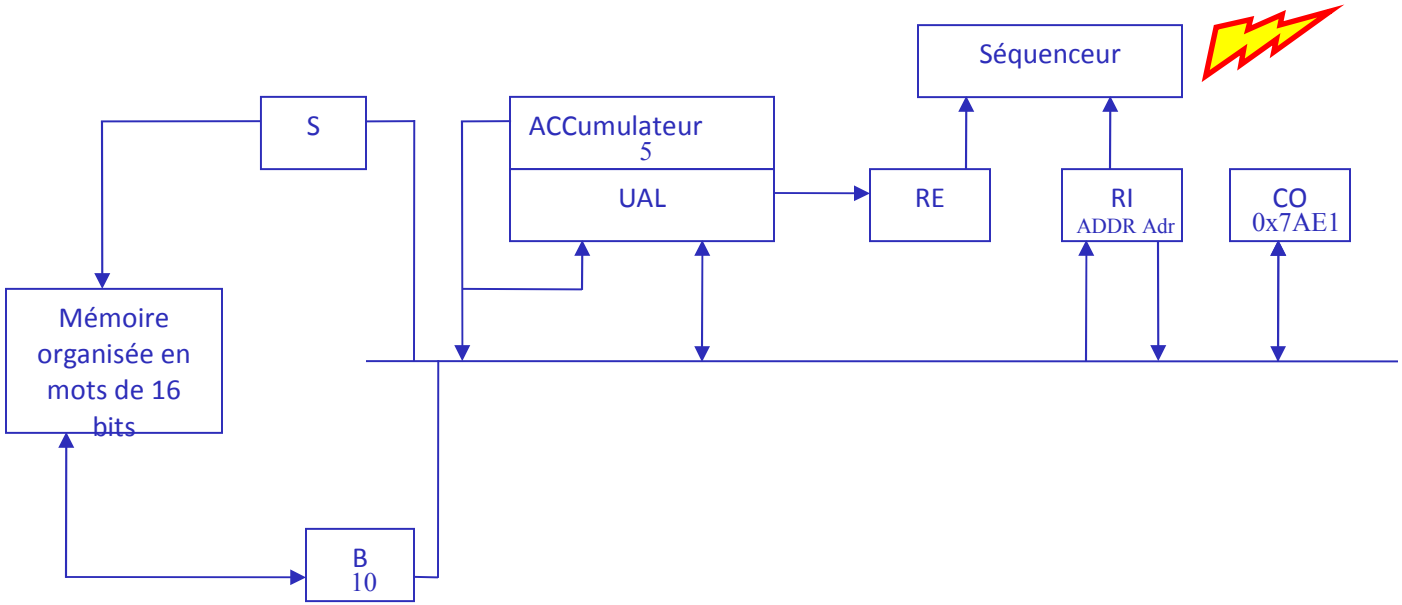
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



16- Le contrôleur de la mémoire délivre un signal de validation

Exemple d'un processeur 16 bits à 1 bus

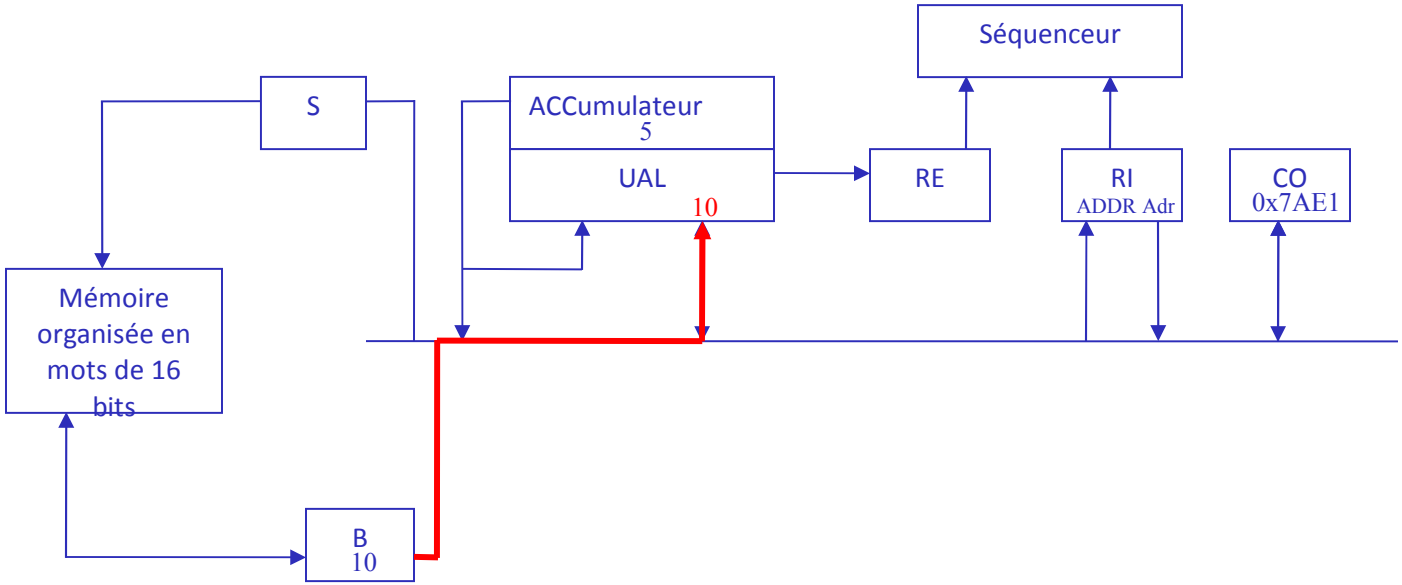
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



17- Le séquenceur délivre un signal

Exemple d'un processeur 16 bits à 1 bus

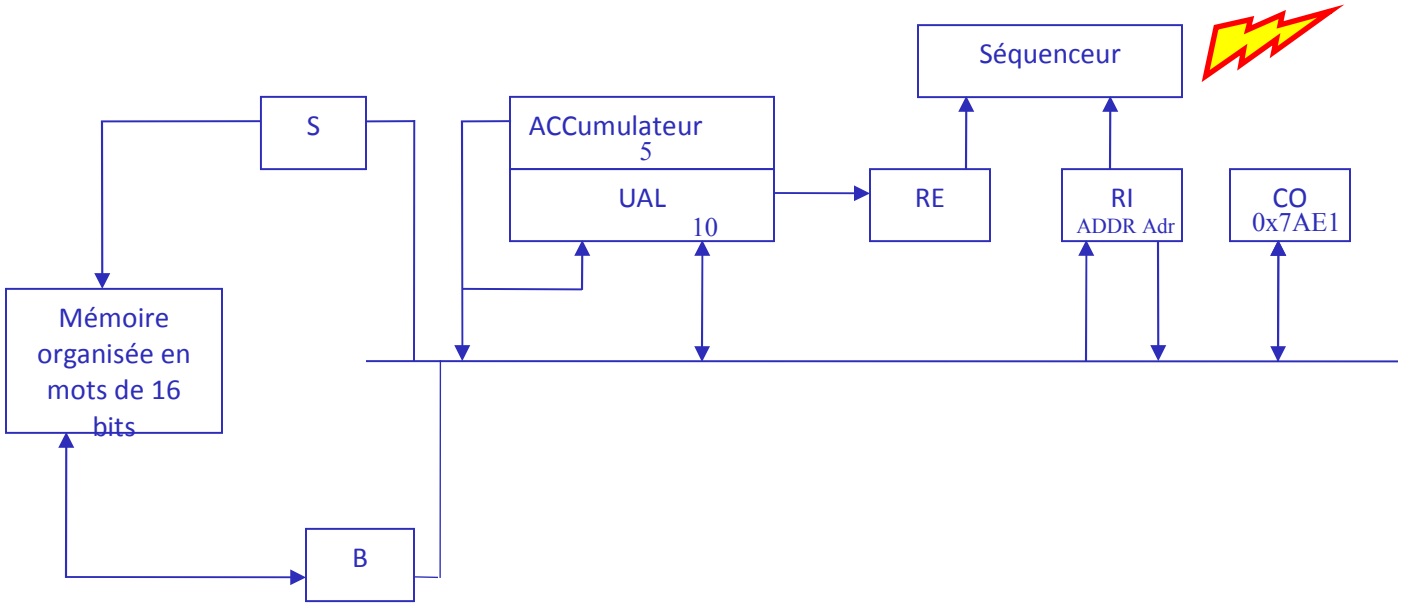
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



18- La donnée est placée dans la première entrée de l'UAL :
(Entrée 1 de l'UAL) := (B)

Exemple d'un processeur 16 bits à 1 bus

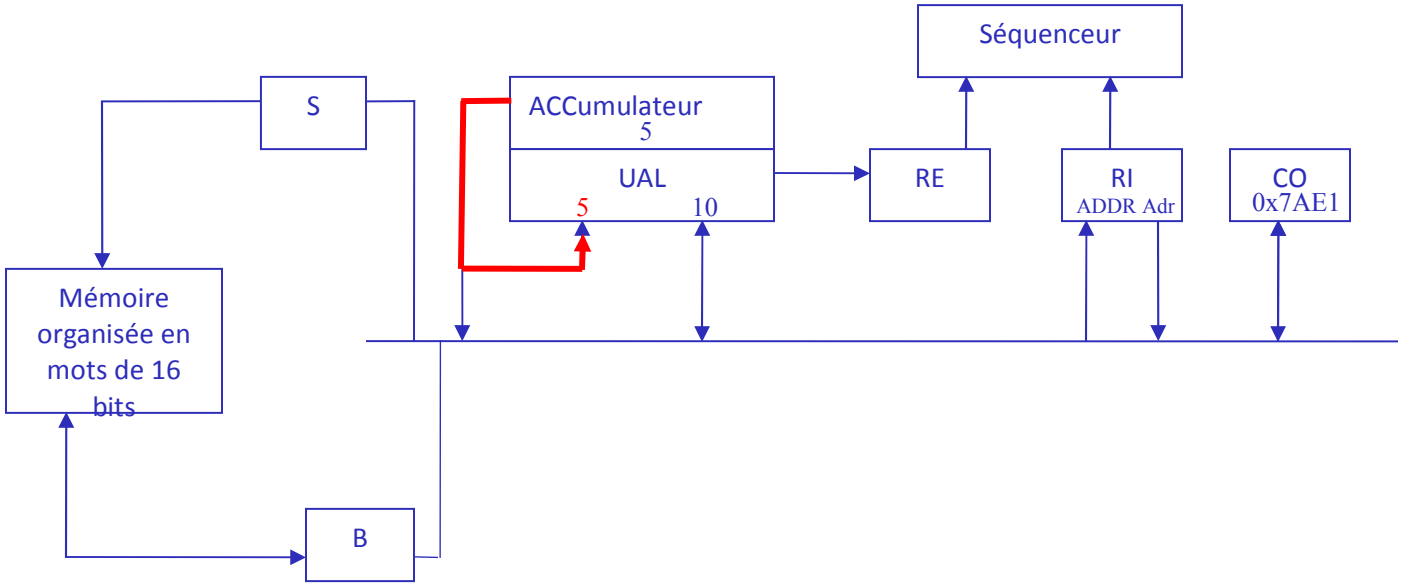
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



19- Le séquenceur délivre un signal

Exemple d'un processeur 16 bits à 1 bus

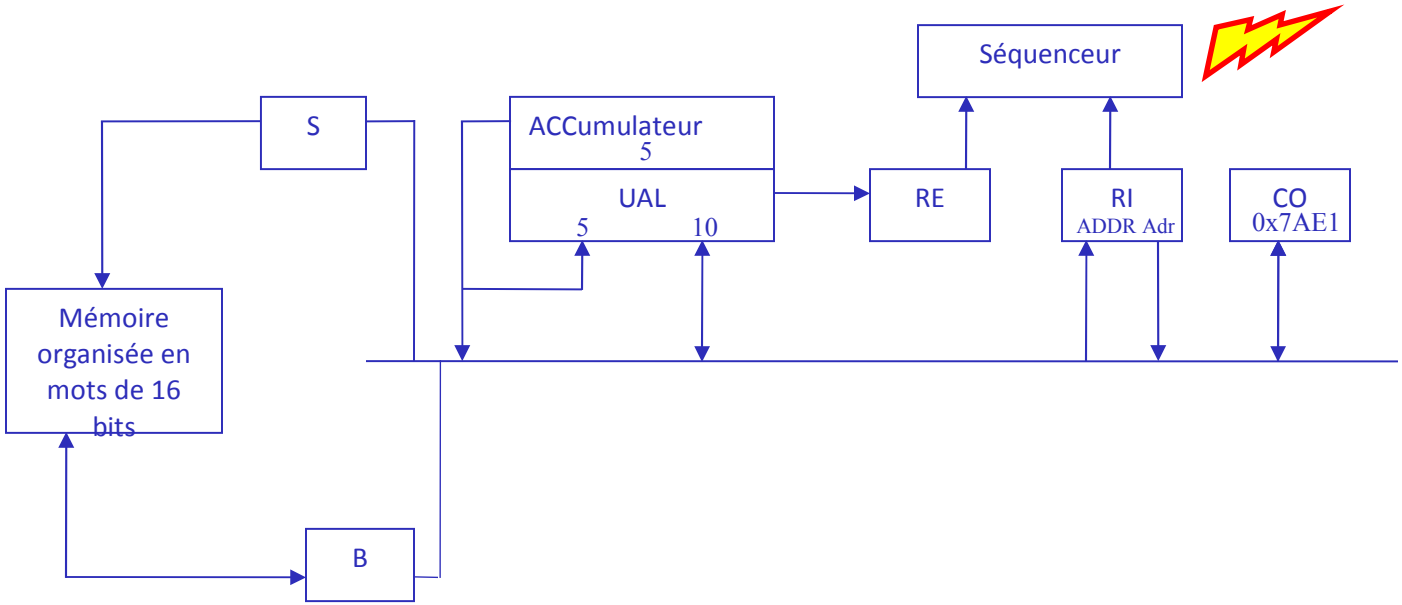
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



20- Le contenu de l'accumulateur est placé dans la deuxième entrée de l'UAL : (Entrée 2 de l'UAL) := (ACC)

Exemple d'un processeur 16 bits à 1 bus

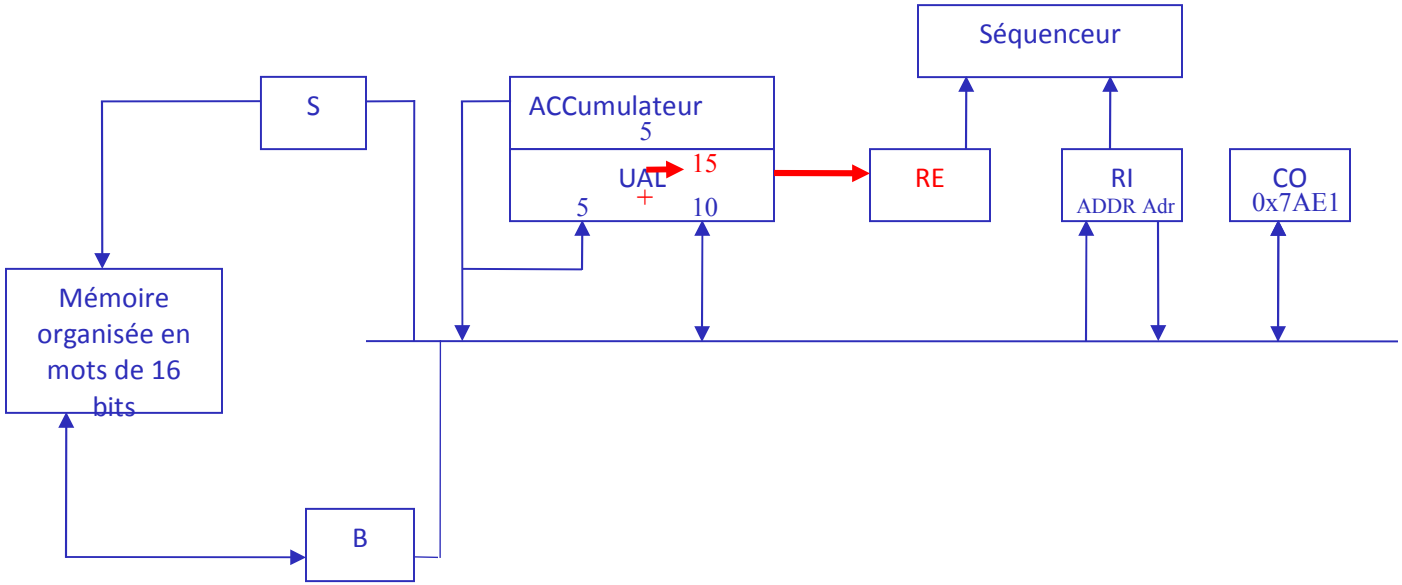
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



21- Le séquenceur délivre un signal d'opération

Exemple d'un processeur 16 bits à 1 bus

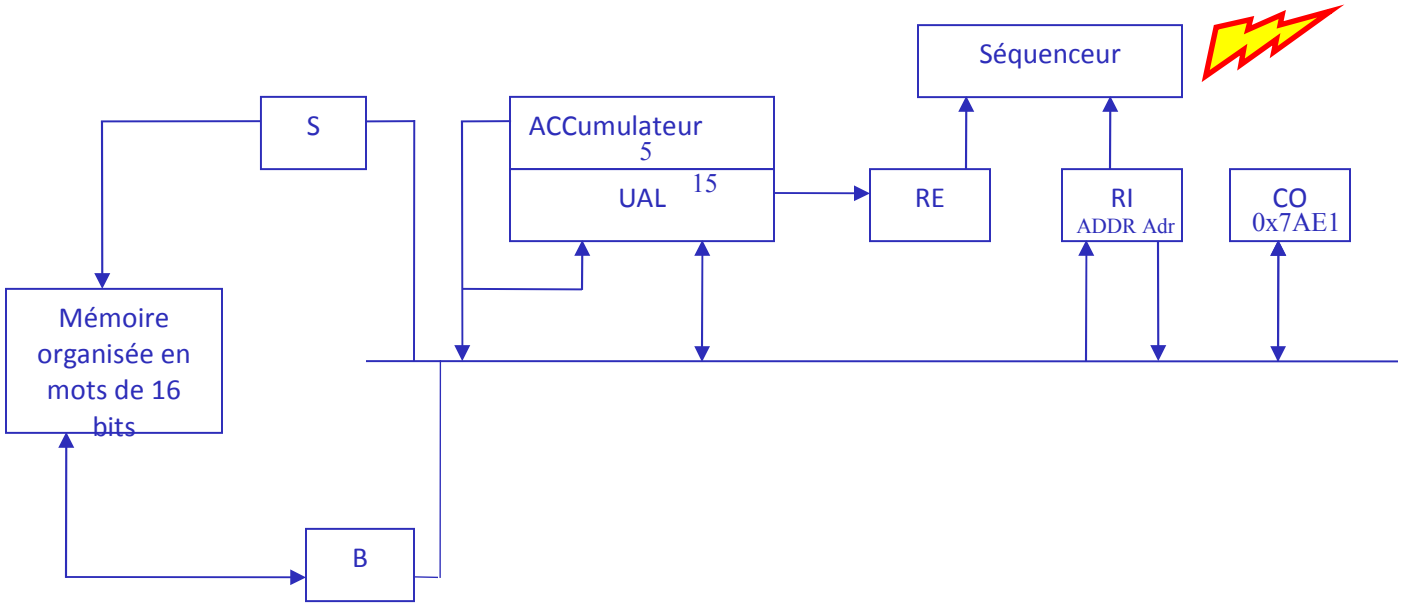
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



21- L'opération est exécutée.

Exemple d'un processeur 16 bits à 1 bus

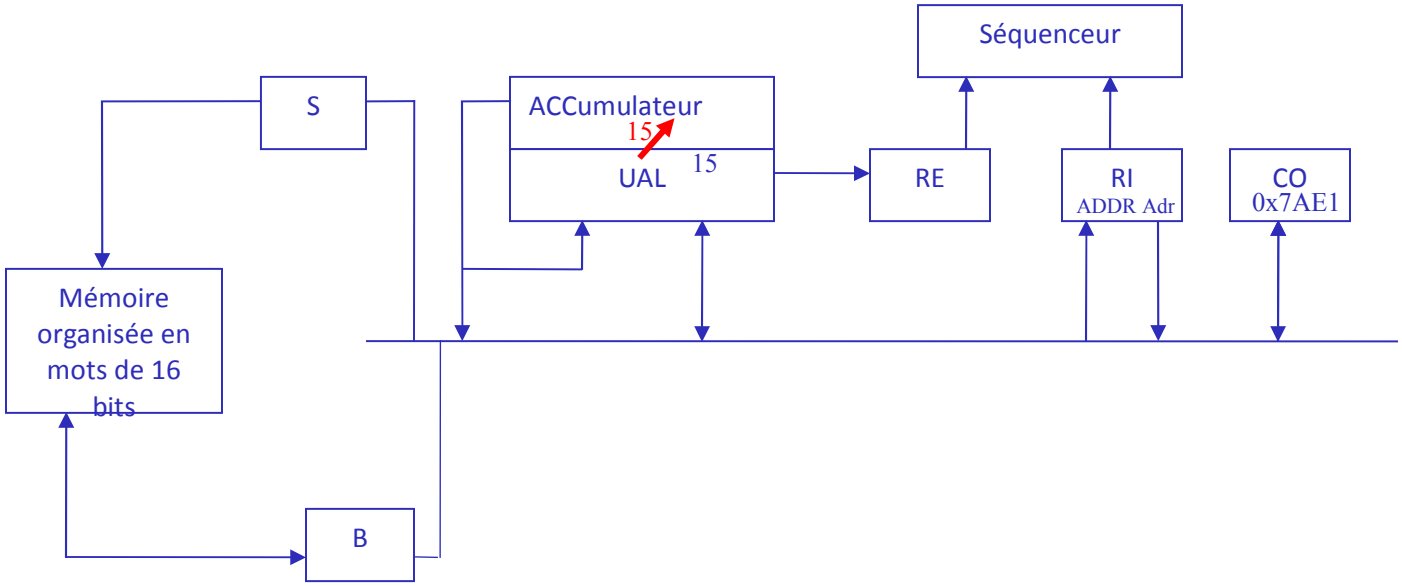
Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



22- Le séquenceur délivre un signal

Exemple d'un processeur 16 bits à 1 bus

Trace de l'exécution de l'instruction ADDR adr
(ACC) := (ACC) + (Adr)



23- Le résultat de l'opération est placé dans l'accumulateur :
(ACC) := (Sortie UAL)

Exemple d'un processeur 8 bits à 1 bus

Exercice

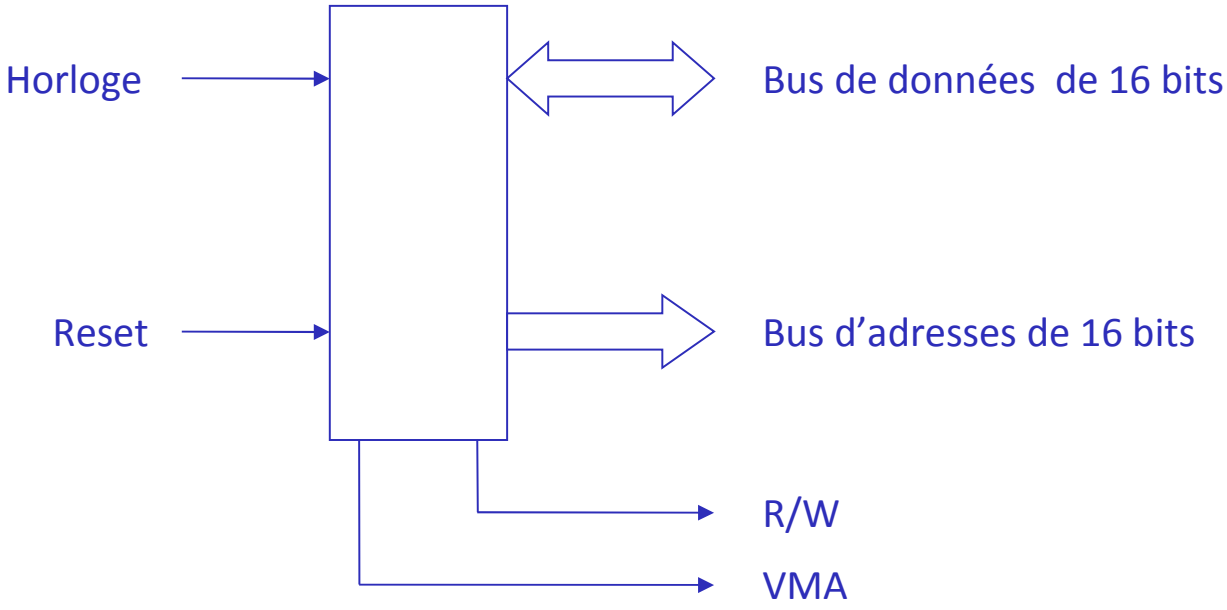
On suppose que le bus n'a que 8 bits, que la mémoire est organisée en octets et chaque instruction est codée sur deux octets. Tracer l'exécution complète des instructions suivantes :

ADD Adr ; (ACC) := (ACC) + (Adr)

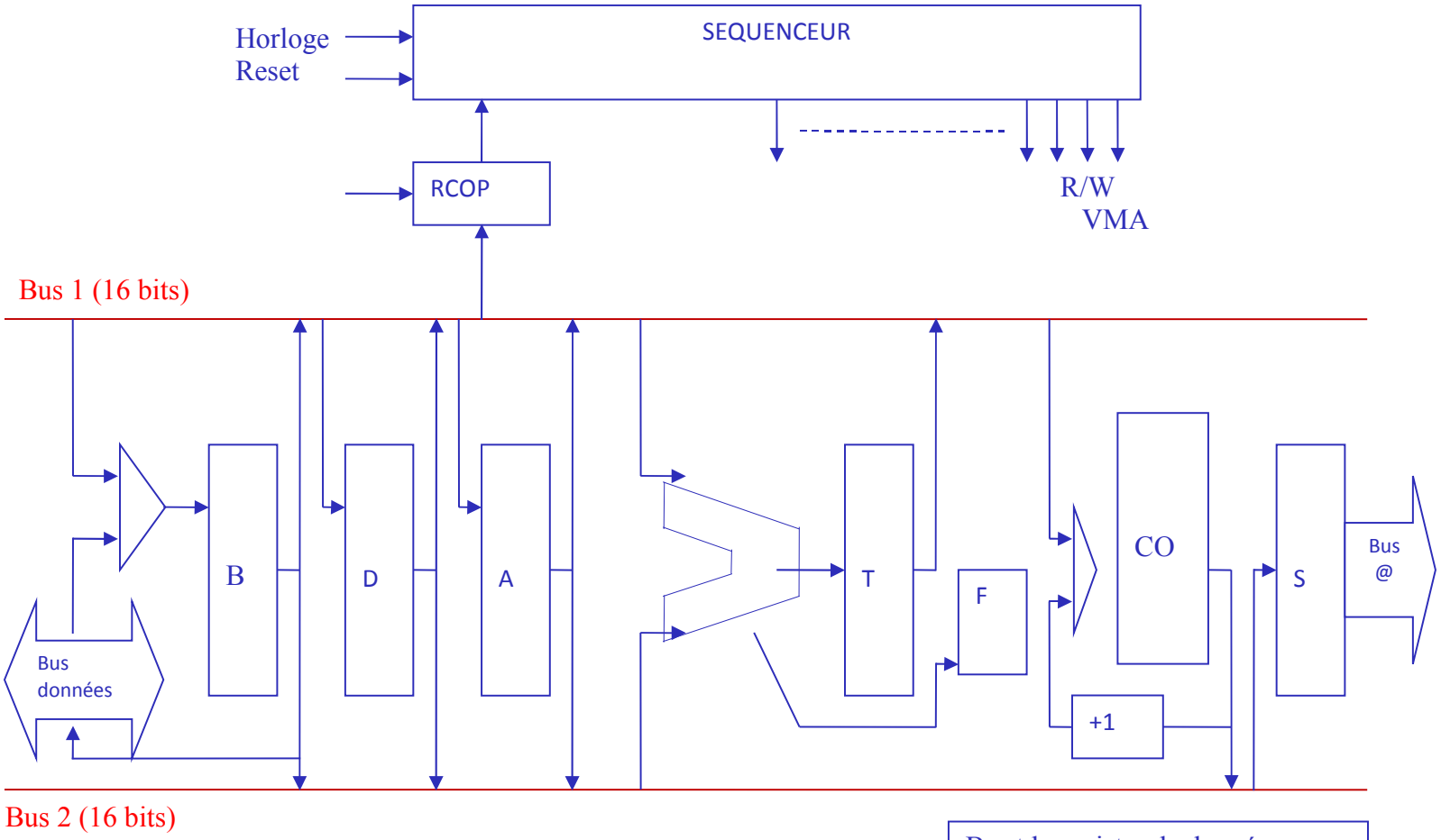
RAN Adr ; (Adr) := (ACC)

SAUT Adr ; (CO):=ADR

Exemple d'un processeur 16 bits à 2 bus



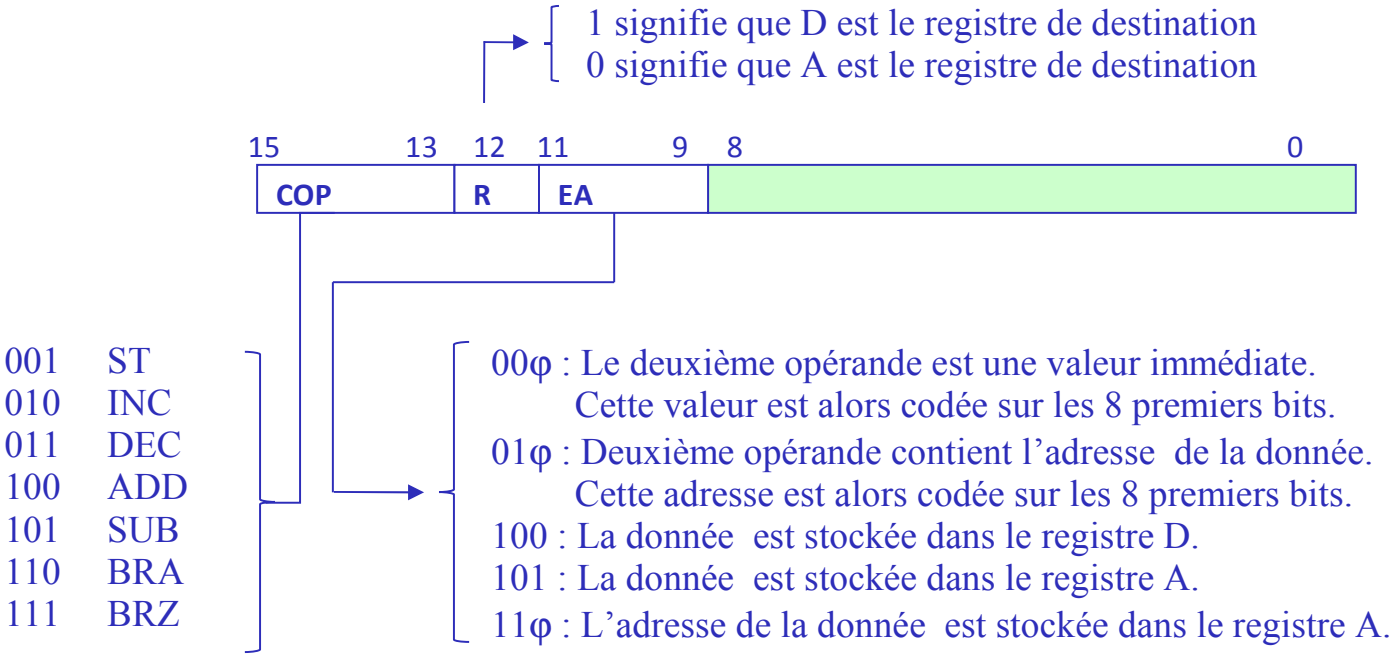
Exemple d'un processeur 16 bits à 2 bus



B est le registre de données
A et D sont des accumulateurs
T est le registre temporaire
S registre d'adresses

Exemple d'un processeur 16 bits à 2 bus

Les instructions de cette machine sont codées de la manière suivante

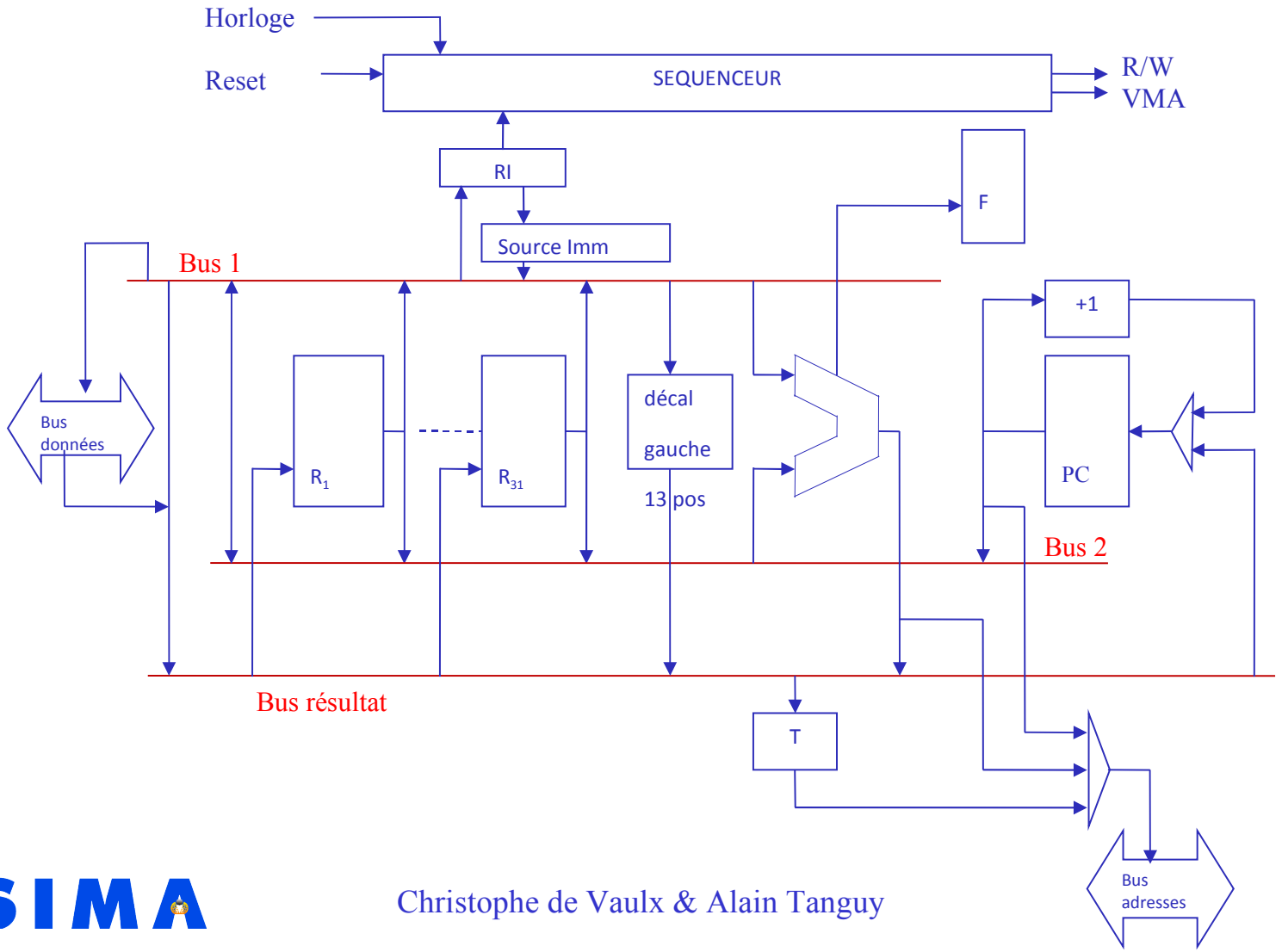


Exemple d'un processeur 16 bits à 2 bus

Avantage par rapport au processeur à un bus :

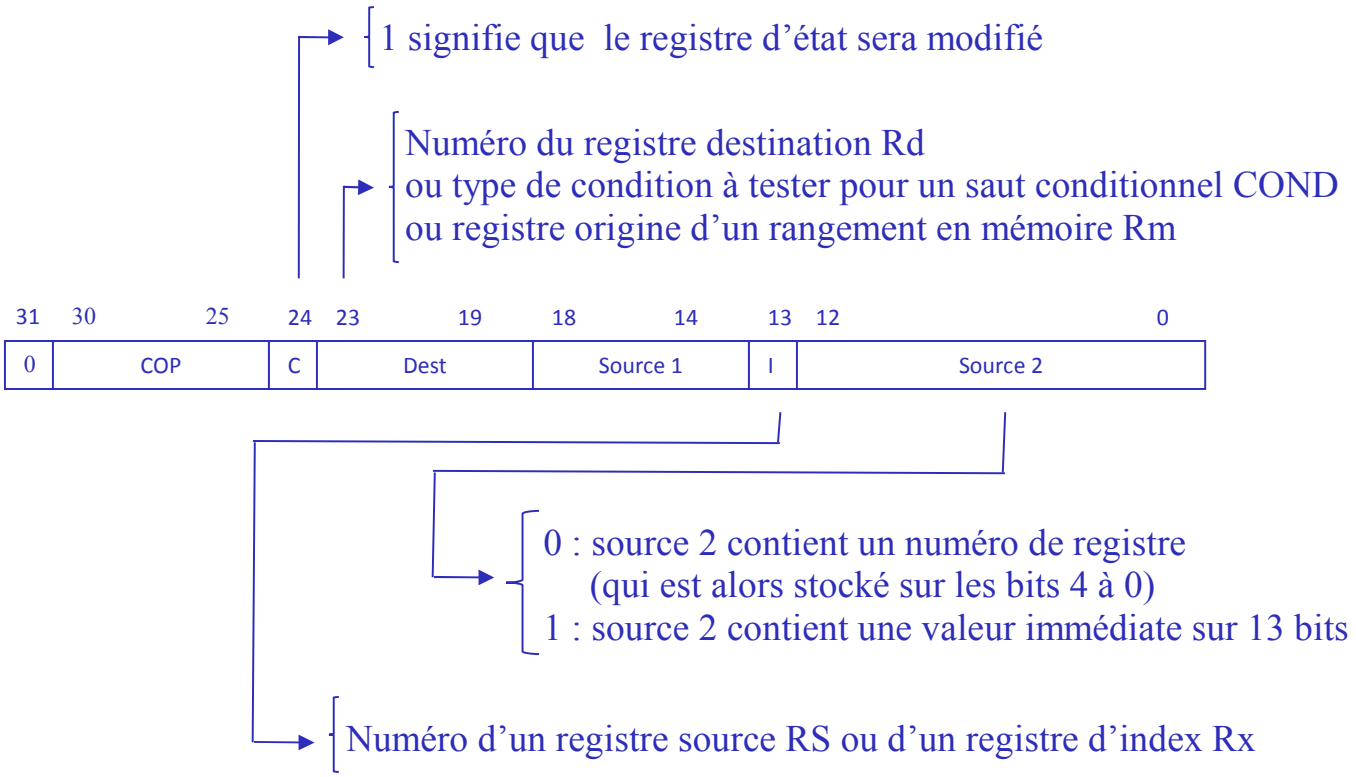
Le fait d'avoir deux bus permet d'effectuer des écritures mémoire, des transferts de données et d'adresses en parallèle et donc d'aller plus vite…

Exemple d'un processeur RISC 32 bits à 3 bus



Exemple d'un processeur RISC 32 bits à 3 bus

Les instructions de cette machine sont codées de la manière suivante



Exemple d'un processeur RISC 32 bits à 3 bus

Jeu d'instruction de la machine

COP	Instruction	Opérandes	Description	
000000	ADD	Rs, S2, Rd	Rd ← Rs + S2	Addition
000001	SUB	Rs, S2, Rd	Rd ← Rs - S2	Soustraction 1
000010	SUBR	Rs, S2, Rd	Rd ← S2 – Rs	Soustraction 2
000011	AND	Rs, S2, Rd	Rd ← Rs ET S2	ET logique
000100	OR	Rs, S2, Rd	Rd ← Rs OU S2	OU logique
000101	XOR	Rs, S2, Rd	Rd ← Rs XOR S2	Disjonction
001000	LD	(Rx)S2, Rd	Rd ← M(Rx+S2)	Chargement
001011	LDHI	Rd, Y	Rd31,13 ← Y Rd12,0 ← 0	Chargement immédiat poids fort
001001	ST	Rm,(Rx)S2	M(Rx+S2) ← Rm	Rangement
010000	JMP	COND, S2(Rx)	PC ← Rx + S2	Saut cond indexé
010001	JMPR	COND, Y	PC ← PC + Y	Saut cond relatif

Exemple d'un processeur RISC 32 bits à 3 bus

Avantage supplémentaire par rapport au processeur à deux bus :

Cela permet de charger les données dans l'UAL et de ranger le résultat dans un registre en parallèle donc d'aller plus vite…

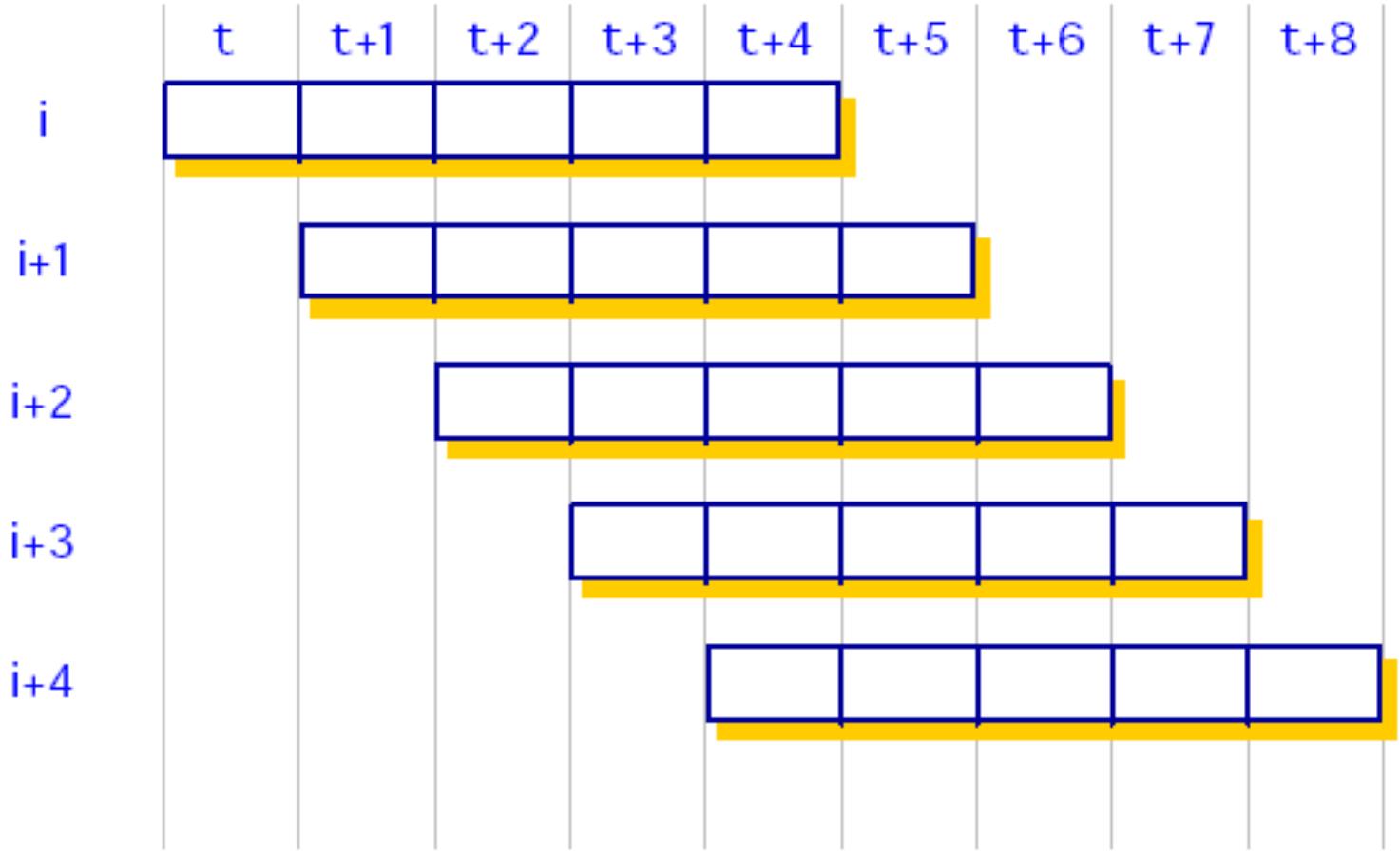
Cela facilite aussi la gestion d'un pipeline !!!

ranger le résultat d'une instruction //
charger les données de la suivante

Pipeline

- Technique utilisée pour optimiser le temps d'exécution d'une instruction.
- Si le temps d'exécution d'une instruction est T_i, l'exécution séquentielle de m instructions prend un temps T_t : $T_t = m * T_i$
- Si l'instruction est décomposée en n étapes, chacune d'une durée T_s, alors $T_t = (m + n - 1) * T_s < m * n * T_s$
- Le principe du pipeline est d'affecter à chaque étape une ressource indépendante.
- Ainsi, on peut exécuter plusieurs instructions en parallèle, chacune à une étape différente.

Pipeline



Architecture externe à base de Pentium