Architecture des ordinateurs

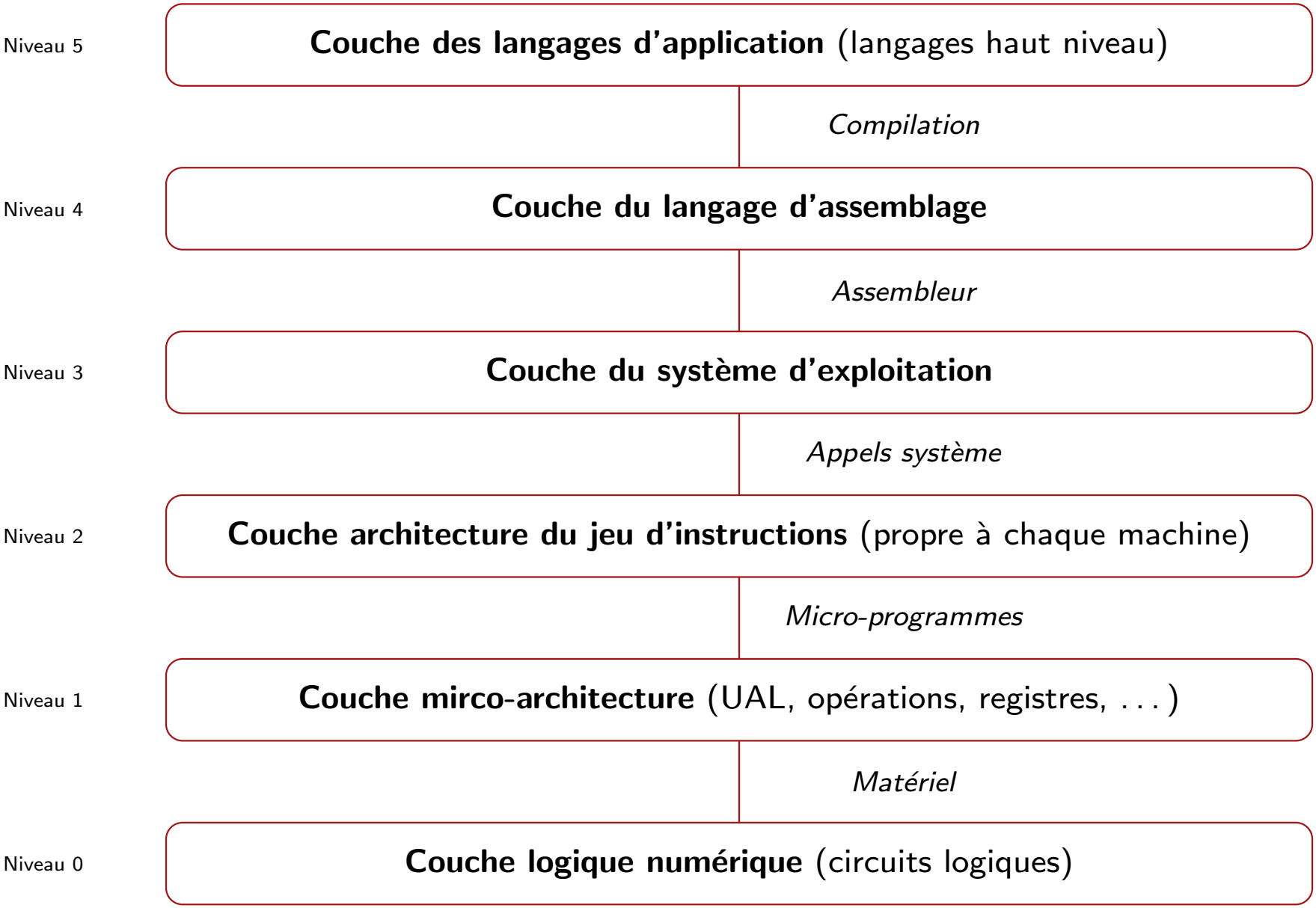
La couche ISA
(*Instruction Set Architecture*)

L1 MIASHS

UFR Mathématiques et Informatique

(2014 - 2015)

Rappel : architecture en couches



Architecture du jeu d'instructions (1/2)

- Définit l'**architecture fonctionnelle** de l'ordinateur
- Sert d'**interface** entre les couches logicielles et le matériel sous-jacent
- Définit le jeu d'instructions utilisables pour coder les programmes :
  - Directement implémenté de façon matérielle
    $\rightarrow$ pas de registre d'état interne servant de compteur ordinal
  - Implémenté sous forme micro-programmée

Architecture du jeu d'instructions (2/2)

Le jeu d'instructions est **indépendant de considérations d'implémentation** :

- Liberté d'implémentation en fonction :
  - des coûts de conception et de fabrication
  - de la complexité de réalisation
  - du coût souhaité
- Nécessité pour le compilateur de connaître l'implémentation de la machine cible pour générer du code efficace

RISC *vs.* CISC

CISC (*Complex Instruction Set Computer*)

- Jeu étendu d'instructions complexes
- 1 instruction peut effectuer **plusieurs opérations élémentaires**

  (ex : charger une valeur en mémoire, faire une opération arithmétique et ranger le résultat en mémoire)
- Instructions proches des constructions typiques des langages haut-niveau
- Exemples : x86 Intel, AMD, ...

RISC (*Reduced Instruction Set Computer*)

- Jeu d'instructions réduit
- 1 instruction effectue **une seule opération élémentaire** (micro-instruction)
- Plus uniforme (même taille, s'exécute en un cyle d'horloge)
- Exemples : PowerPC, UltraSPARC, ...

Format d'une instruction

- *opcode* : code le type d'opération réalisée par l'instruction (obligatoire)
- Autres champs (optionnels) :
  - Spécifient les adresses des opérandes de l'instruction
  - Entre 0 et 3 adresses
- Soit toutes de la même taille, soit de tailles différentes :
  - Même taille $\Rightarrow$ décodage simplifié mais conso. mémoire plus importante
  - Mot mémoire $\leqslant$ Taille instruction ou Taille instruction $\leqslant$ Mot mémoire

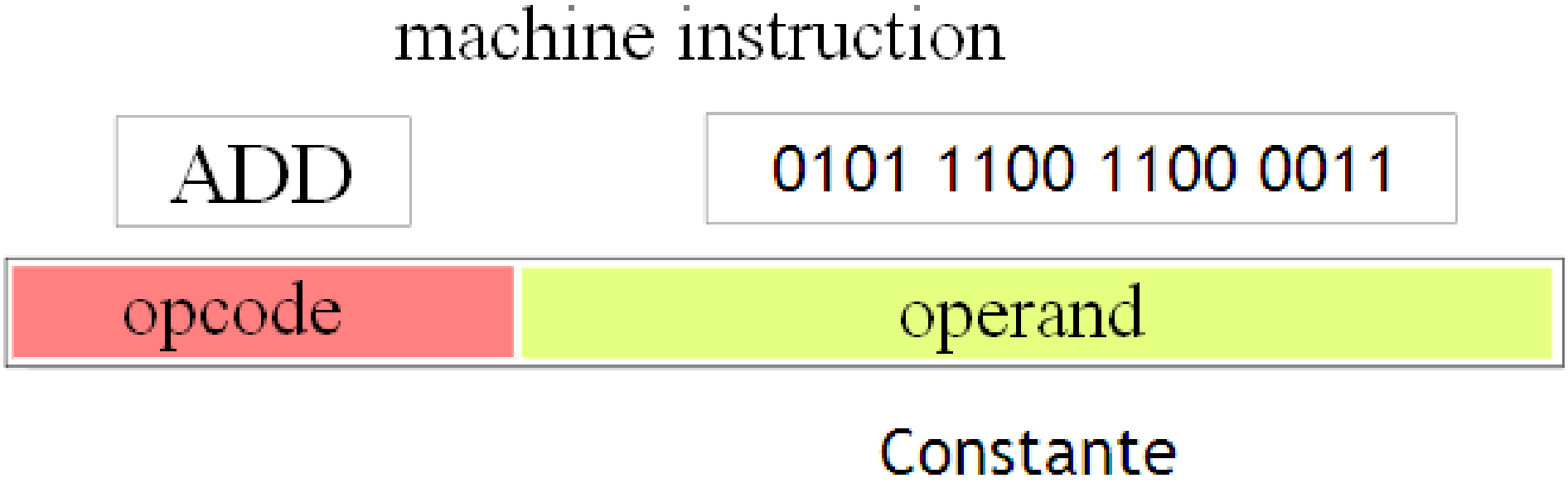
Modes d'adressage

- Différentes manières dont on peut accéder aux opérandes des instructions
- Pour les programmeurs en assembleur et les auteurs de compilateurs
- Principaux modes :
  - Adressage immédiat
  - Adressage direct
  - Adressage registre
  - Adressage indirect par registre
  - Adressage indexé
  - Adressage basé indexé

Adressage immédiat (1/2)

- Adressage “le plus simple”
- La partie adresse de l'instruction contient directement la valeur de l'opérande
- Réservé aux constantes
- Aucun accès mémoire supplémentaire nécessaire
- Exemples :
  - Branchements
  - Chargement de registres

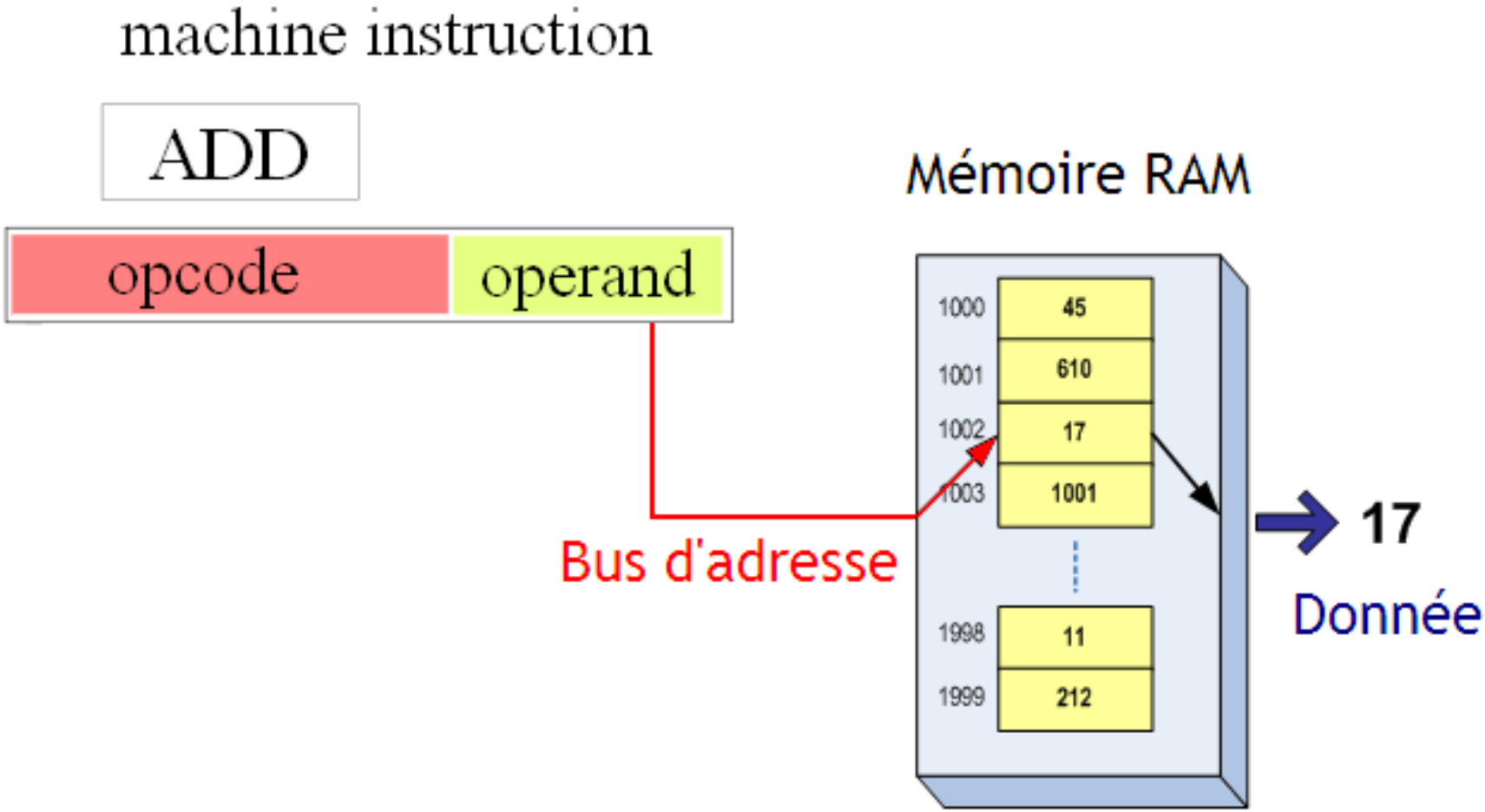
Adressage immédiat (2/2)



Adressage direct (1/2)

- Adresse fournie pour un accès direct
- Accès toujours à la même zone mémoire
- Permet de lire ou d'écrire une donnée sans la copier avant dans un registre
- Réservé aux variables globales (adresses connues à la compilation)

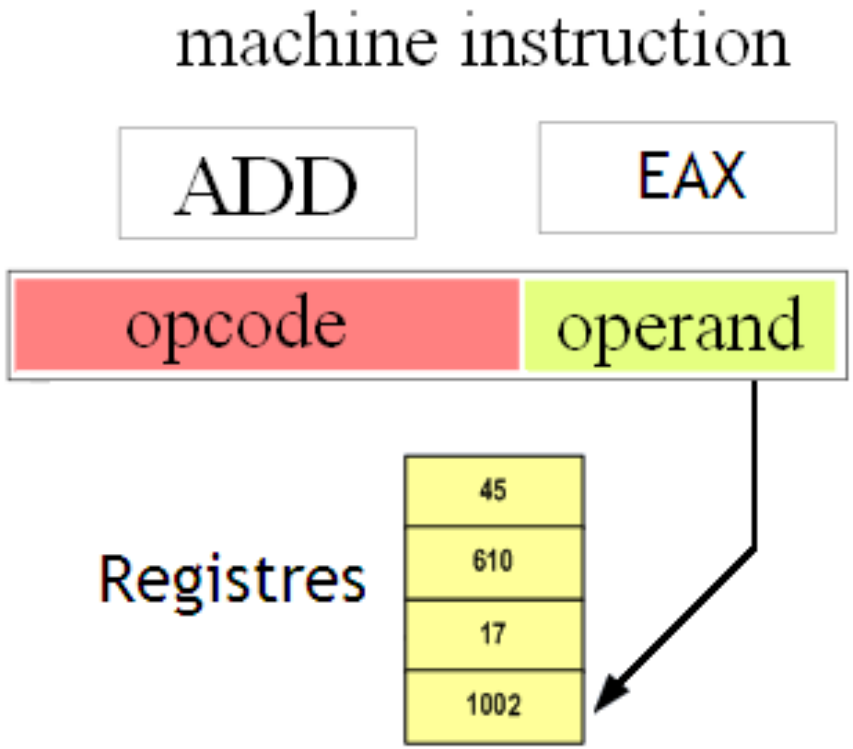
Adressage direct (2/2)



Adressage registre (ou inhérent) (1/2)

- Équivalent à l'adressage direct
- Spécification d'un numéro de registre plutôt qu'un numéro de mot mémoire
- Mode le plus couramment utilisé :
  - Accès aux registres très rapides
  - Numéros de registres codés sur peu de bits
- Une grande partie du travail des compilateurs consiste à déterminer quelles variables seront placées dans quel regsitres à chaque instant, afin de diminuer les temps d'accès et donc d'exécution.

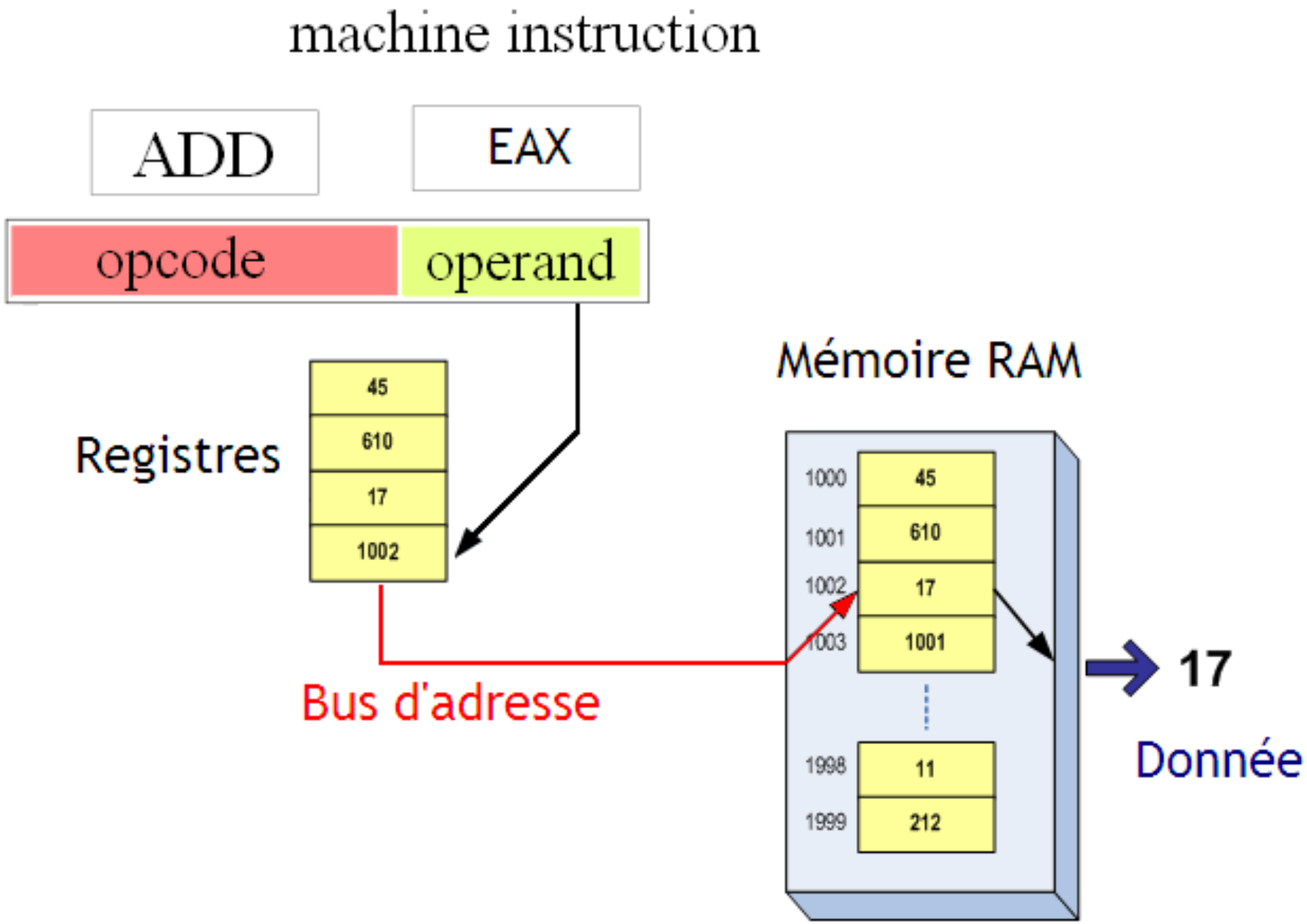
Adressage registre (ou inhérent) (2/2)



Adressage indirect par registre (1/2)

- L'opérande spécifié provient de la mémoire
- Adresse de l'opérande contenue dans un registre de numéro donné
  - Le registre est un pointeur sur l'opérande
  - Référencement d'une zone mémoire sans avoir à coder son adresse dans l'instruction
  - Modification dynamique de l'adresse de la zone mémoire référencée en modifiant la valeur du registre

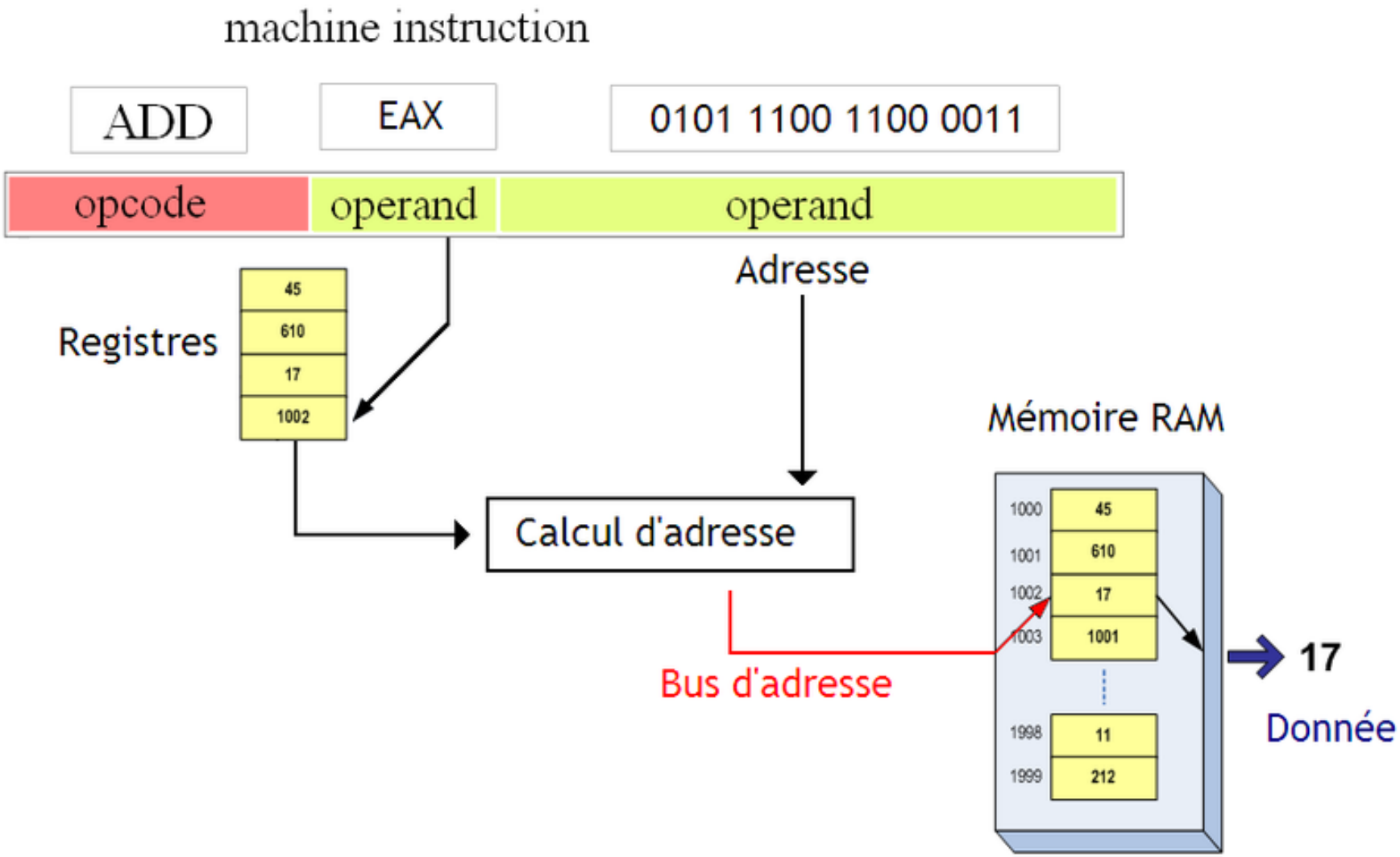
Adressage indirect par registre (2/2)



Adressage indexé (1/2)

- Combine les caractéristiques de l'adressage direct et de l'adressage registre
- Localisation à une distance fixe de l'adresse fournie par un registre
- Les champs de l'instruction sont :
  - Numéro du registre
  - Déplacement relatif (*offset*) à ajouter à son contenu
- Ex : Accès aux variables locales et paramètres dans la pile par rapport à BP

Adressage indexé (2/2)

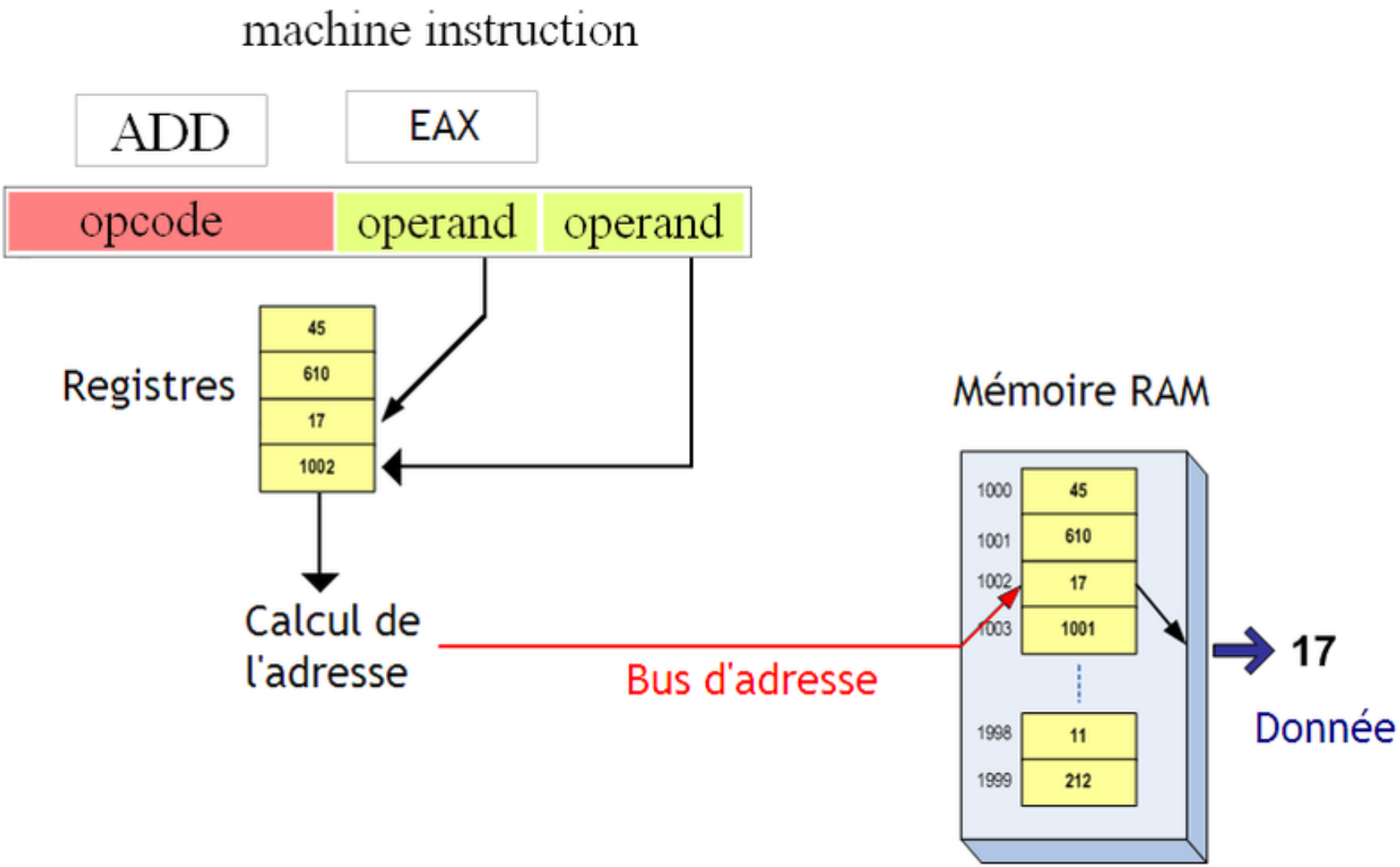


Adressage basé indexé (1/2)

- Adresse calculée à partir de :
  - La somme des valeurs de deux registres ;
    - ★ Un registre de base
    - ★ Un registre d'index
  - Une valeur de déplacement optionnelle

- Ex : Accès aux champs des structures contenues dans un tableau
  - Le registre de base est l'adresse de début du tableau
  - Le registre d'index référence l'adresse de début de la bonne structure par rapport à l'adresse du tableau
  - Le déplacement référence la position du début du champ par rapport au début de la structure

Adressage basé indexé (2/2)



Types d'instructions

- Copie de données
- Calcul
- Branchements, branchements conditionnels et comparaisons
- Entrées / Sorties et interruptions
- Gestion de la mémoire

Instructions de copie de données

- Deux usages principaux :
  - Réaliser l'affectation de valeurs à des variables

  → Recopie de valeurs dans des variables temporaires pour des calculs ultérieurs
  - Placer une copie de valeurs utiles là où elles pourront être accédées le plus efficacement

  → Utilisation des registres plutôt que de la mémoire
- Toujours des instructions de copie :
  - entre registres
  - entre registre et mémoire
- Moins souvent de mémoire à mémoire

Instructions de calcul (1/2)

- Représentent les opérations réalisables par l'UAL
- Sur des opérandes qui ne sont pas nécessairement des registres
- Format abrégé pour les plus couramment utilisées
- Ex : Instruction `INC R1` remplace la séquence `MOV R2,1` et `ADD R1,R2`

Instructions de calcul (2/2)

- Dans une architecture de type *load/store*, les seules instructions pouvant accéder à la mémoire sont les instructions `load` et `store` de copie entré mémoire et registre.

- Les instructions de calcul ne prennent dans ce cas que des opérandes registres
  - Simplification du format et du décodage des instructions
  - Permet d'optimiser l'utilisation de l'UAL (pas de cycles d'attente des opérandes mémoires)

Instructions de branchements

- Altèrent le déroulement normal du programme
- Deux catégories :
  - Les sauts : branchement du programme vers une autre adresse mémoire
  - Les sous-routines :
    - Branchement du programmes vers une autre adresse mémoire
    - Sauvegarde de l'adresse située après l'instruction (pour le retour de fonction)
- Peuvent être conditionnels ou inconditionnels

Instructions d'entrées/sorties

- Diffèrent considérablement selon l'architecture
- Mettent en œuvre un ou plusieurs schémas d'E/S différents :
  - E/S programmées avec attente de disponibilité :
    - ★ Très coûteux car le processeur ne fait rien en attendant
    - ★ Cas des instructions `IN` et `OUT` de l'architecture x86
  - E/S par interruptions
    - ★ Interruption envoyée au processeur chaque fois que l'état de l'E/S change
    - ★ Coûteux
  - E/S par DMA (*Direct Memory Access*) : un circuit spécialisé se charge des échanges de données

Instructions d'interruption

- Conduisent à l'exécution d'une routine de traitement adaptée
  - Exécution suspendue du programme en cours
  - Exécution de la routine de traitement

  $\Rightarrow$ Analogue à un appel de sous-programme, mais de façon asynchrone

- Plusieurs types d'interruptions identifiées par un numéro

- Peuvent êtres :
  - Asynchrones
    - ⋆ Interruptions matérielles
    - ⋆ Par activation des certaines des lignes de contrôle du processeur
  - Synchrones
    - ⋆ Générées par le processeur lui-même
    - ⋆ Par instruction spécifique (mise en œuvre des appels systèmes)
    - ⋆ Sur erreur logicielle (mise en œuvre des exceptions)

Exécution d'une interruption

1. Sauvegarde dans la pile de la prochaine instruction à exécuter dans le cadre du déroulement normal
2. Utilisation du numéro de l'interruption pour indexer une table contenant les adresses des différentes routines de traitement
3. Déroutage à cette adresse
4. Passage en mode privilégié si le processeur en dispose