

# De l'ordinateur au processus : rôle d'un système

CSC 3102

Introduction aux systèmes d'exploitation

Gaël Thomas



# Présentation du cours

## □ Contexte du cours :

- Introduire notre objet d'étude : les systèmes d'exploitation

## □ Objectifs :

- Comprendre ce qu'est un ordinateur
- Comprendre ce qu'est un système d'exploitation
- Comprendre ce qu'est un processus
- Comprendre ce que sont une application et un logiciel

## □ Notions abordées :

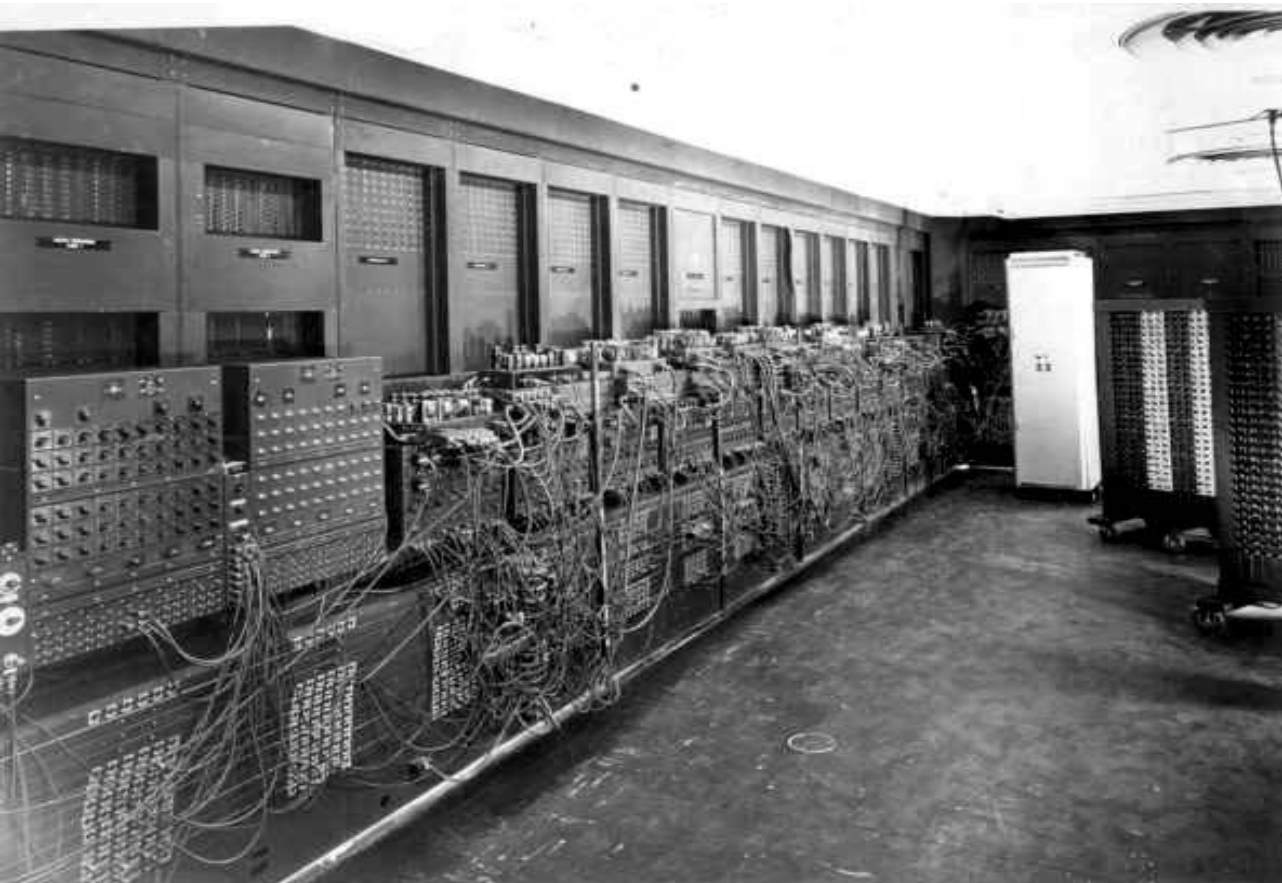
- Ordinateur, mémoire, processeur, périphérique, système d'exploitation, processus, communication, application, logiciel

# I. Qu'est ce qu'un ordinateur ?



# Définition d'un ordinateur

- Machine électronique capable d'exécuter des instructions effectuant des opérations sur des nombres



*1946 : ENIAC  
(calculateur à tubes  
30 tonnes, 72m<sup>2</sup>  
pour 330 mult/s)*

# Définition d'un ordinateur

- Machine électronique capable d'exécuter des instructions effectuant des opérations sur des nombres



*Janv 1948 : SSEC (premier ordinateur chez IBM) avec une capacité mémoire de 150 nombres*

# Définition d'un ordinateur

- Machine électronique capable d'exécuter des instructions effectuant des opérations sur des nombres

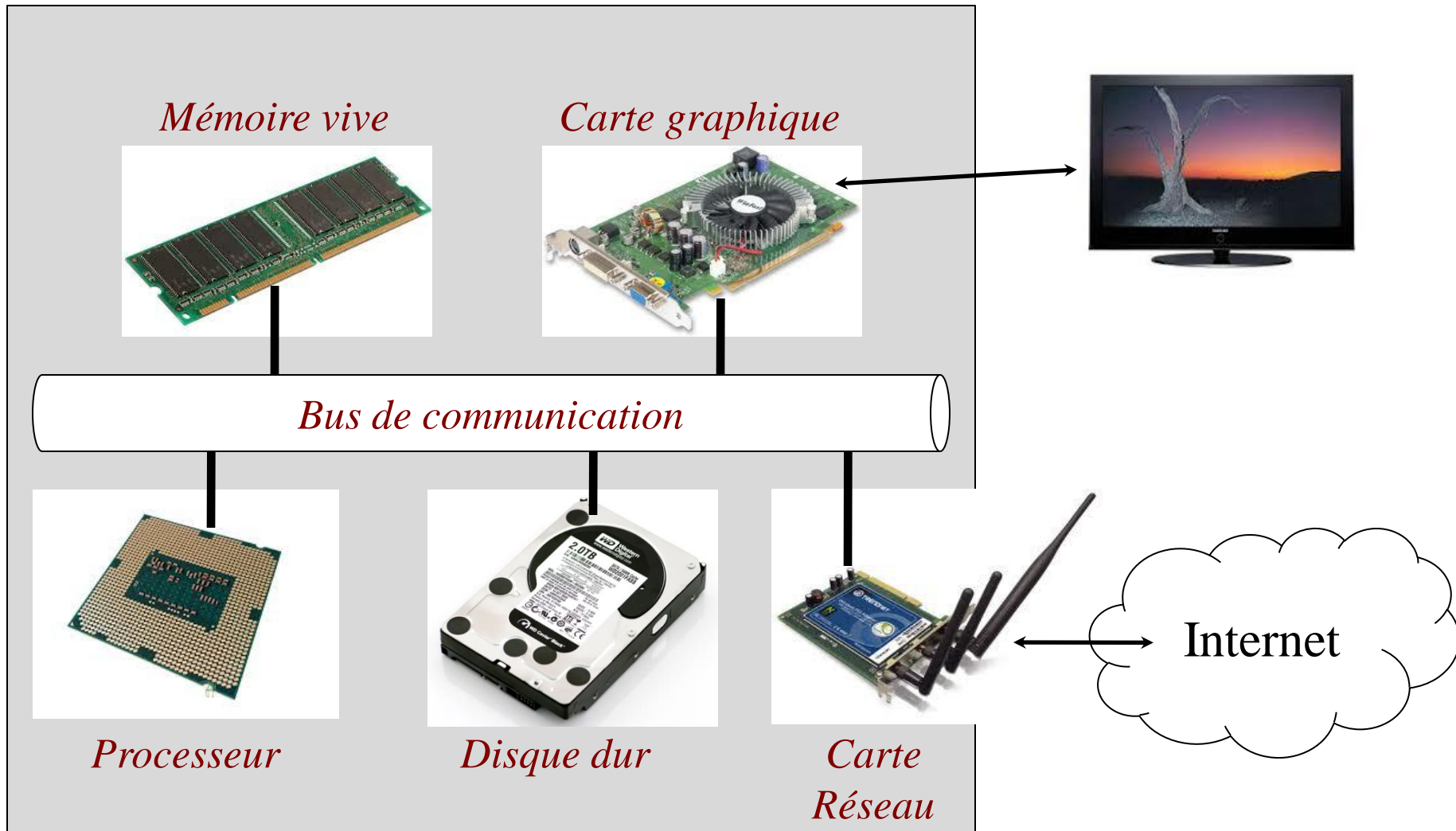


# Schéma de haut niveau d'un ordinateur

- **Processeur** : unité qui s'occupe d'exécuter les instructions
- **Mémoire vive** : support stockant les données de travail du processeur  
Accès rapide, données perdues en cas de coupure électrique.  
Par exemple : SDRAM (Synchronous Dynamic Random Access Memory)
- **Périphériques** : objets fournissant ou stockant des données secondaires  
Réseau, disque dur, souris, clavier, carte graphique, carte son...



# Schéma de haut niveau d'un ordinateur





# Qu'est ce que la mémoire vive

- Mémoire vive : ensemble de cases numérotées contenant des octets
- Une case contient un octet (byte en anglais) = regroupe 8 bits
- Bit : valeur valant 0 ou 1
  - 0 : bit non chargé ("courant ne passe pas")
  - 1 : bit chargé ("courant passe")
- Un octet permet de représenter  $2^8 = 256$  valeurs

Case 0	0110 0001b
Case 1	0101 1001b
Case 2	0110 0001b
Case 3	1111 0000b
	⋮
Case 800	1100 1011b

# Représentation des nombres

- Notation décimale : un chiffre peut prendre 10 valeurs de 0 à 9

$$276 = 2 \cdot 10^2 + 7 \cdot 10^1 + 6 \cdot 10^0$$

- Notation binaire : un chiffre peut prendre 2 valeurs de 0 à 1

$$1101b = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13$$

- Notation hexadécimale : un chiffre peut prendre 16 valeurs de 0 à f

$$0x276 = 2 \cdot 16^2 + 7 \cdot 16^1 + 6 \cdot 16^0 = 630$$

$$0xb6 = 11 \cdot 16^1 + 6 \cdot 16^0 = 182$$

# L'hexadécimal en informatique

- Avec 4 bits, on encode 16 valeurs, soit 1 chiffre hexadécimal
- L'hexadécimal est donc plus concis pour représenter les valeurs des octets
- Un octet est représenté par 2 chiffres hexadécimaux

Case 0	1110 0001b
Case 1	0101 1001b
Case 2	0110 0001b
Case 3	1111 0000b
	⋮
Case 800	1100 1011b

# L'hexadécimal en informatique

- Avec 4 bits, on encode 16 valeurs, soit 1 chiffre hexadécimal
- L'hexadécimal est donc plus concis pour représenter les valeurs des octets
- Un octet est représenté par 2 chiffres hexadécimaux

Case 0	0xe1
Case 1	0x59
Case 2	0x61
Case 3	0xf0
	⋮
Case 800	0xc3

# Que représentent les octets

□ Une série d'octets peut représenter :

- Un entier naturel (dans  $\mathbb{N}$ )
- Un entier relatif (dans  $\mathbb{Z}$ )
- Une suite de caractères
- Une valeur de vérité (vrai ou faux)
- Un nombre flottant
- Un nombre complexe
- Une instruction machine

- Ou tout autre ensemble énumérable

Case 0	0xe1
Case 1	0x59
Case 2	0x61
Case 3	0xf0
	⋮
Case 800	0xc3

# Qu'est ce qu'un processeur

- Un processeur exécute des instructions qui peuvent
  - Effectuer des calculs
  - Accéder à la mémoire
  - Accéder aux autres périphériques
  - Sélectionner l'instruction suivante à exécuter (saut)
- Le processeur identifie une instruction par un numéro (Par exemple : 1 = additionne, 2 = soustrait etc...)

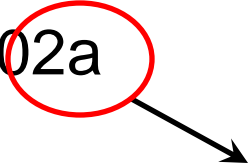
# Ma première instruction

0x4883c02a



# Ma première instruction

0x4883c02a



le nombre 42 encodé en hexadécimal  
( $42 = 2 * 16 + 10$ )

# Ma première instruction

0x4883c02a



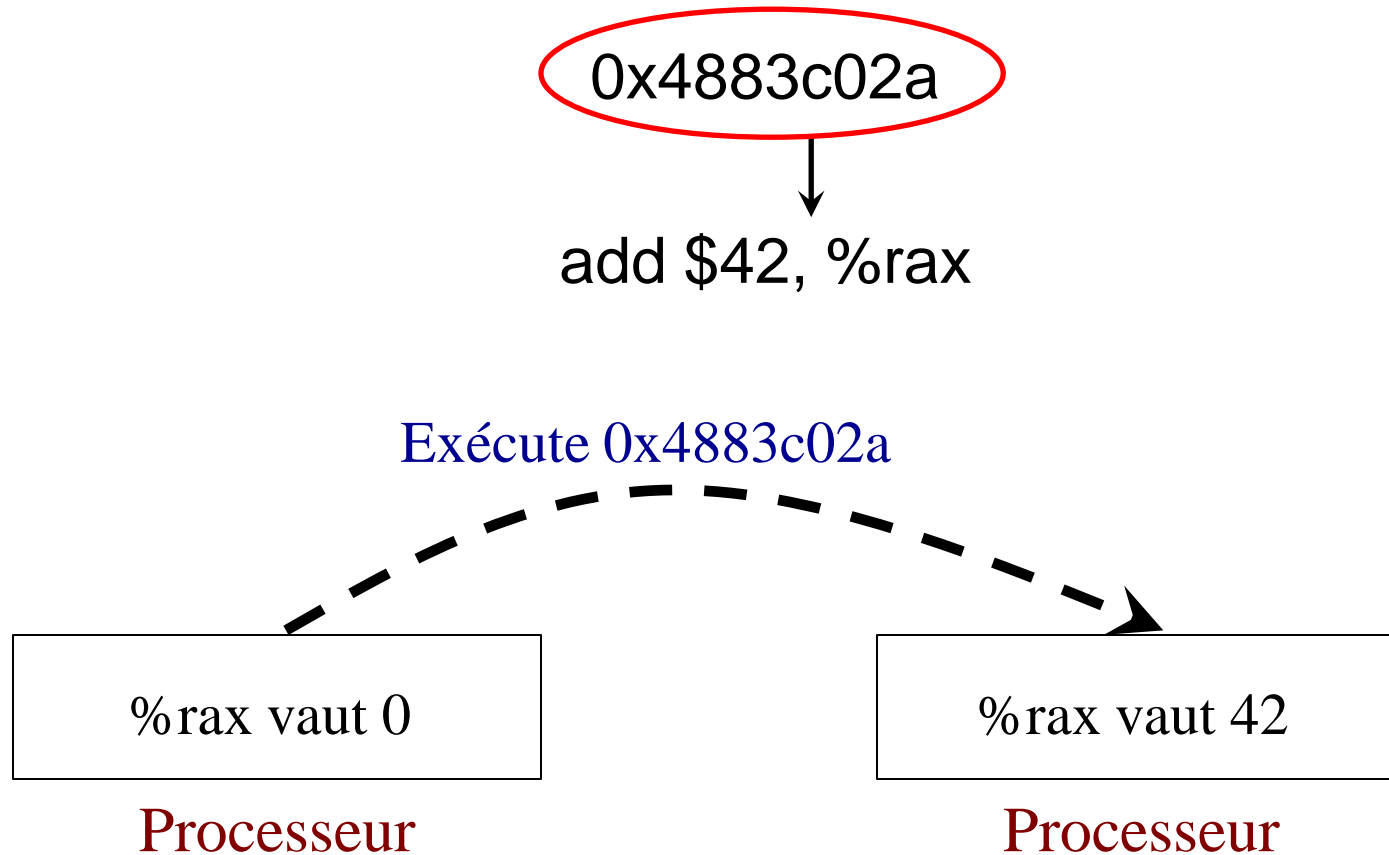
L'instruction d'ajout à **%rax** de l'entier qui suit

%rax = un registre du processeur  
(registre = donnée de travail interne du processeur)

%rax vaut 0

Processeur

# Ma première instruction



Remarque : \$ indique qu'on ajoute 42 et non la valeur contenue dans la case 42 en mémoire

# Ma seconde instruction

0x48890425 0x00000320

800

# Ma seconde instruction

0x48890425 0x00000320

mov %rax, 800

Copie la valeur du registre %rax dans la 800<sup>e</sup> case de la mémoire vive

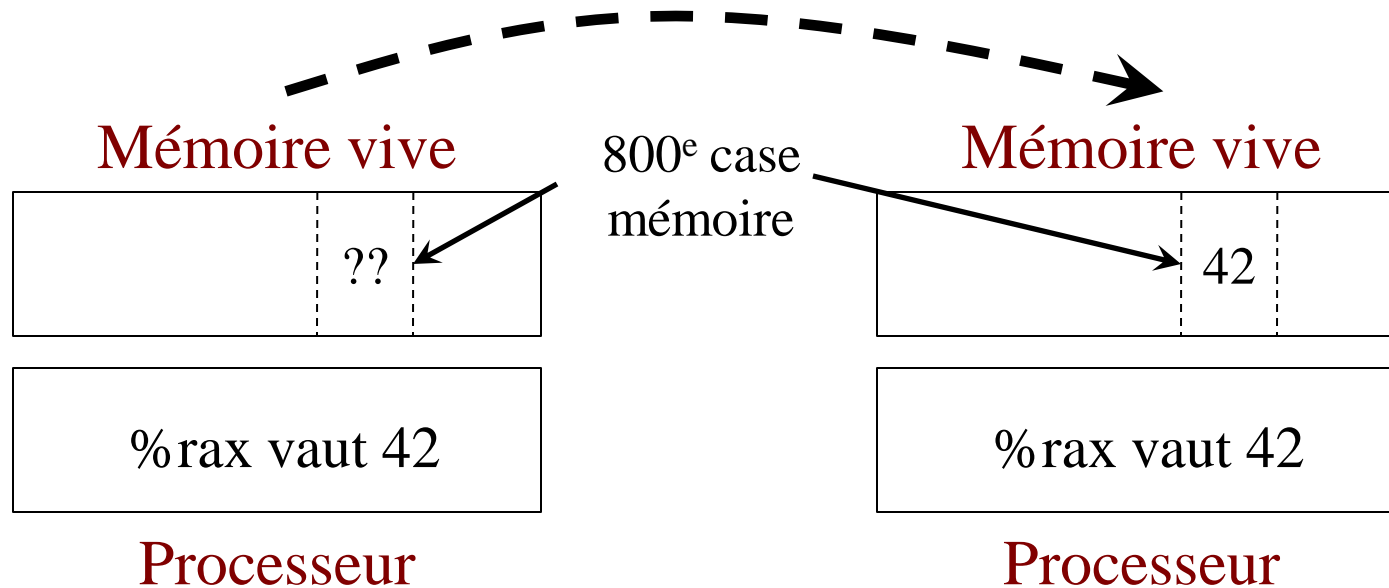
# Ma seconde instruction

0x48890425 0x000000320

mov %rax, 800

Copie la valeur du registre %rax dans la 800<sup>e</sup> case de la mémoire vive

Exécute 0x48890425 0x000000320



# Où sont les instructions

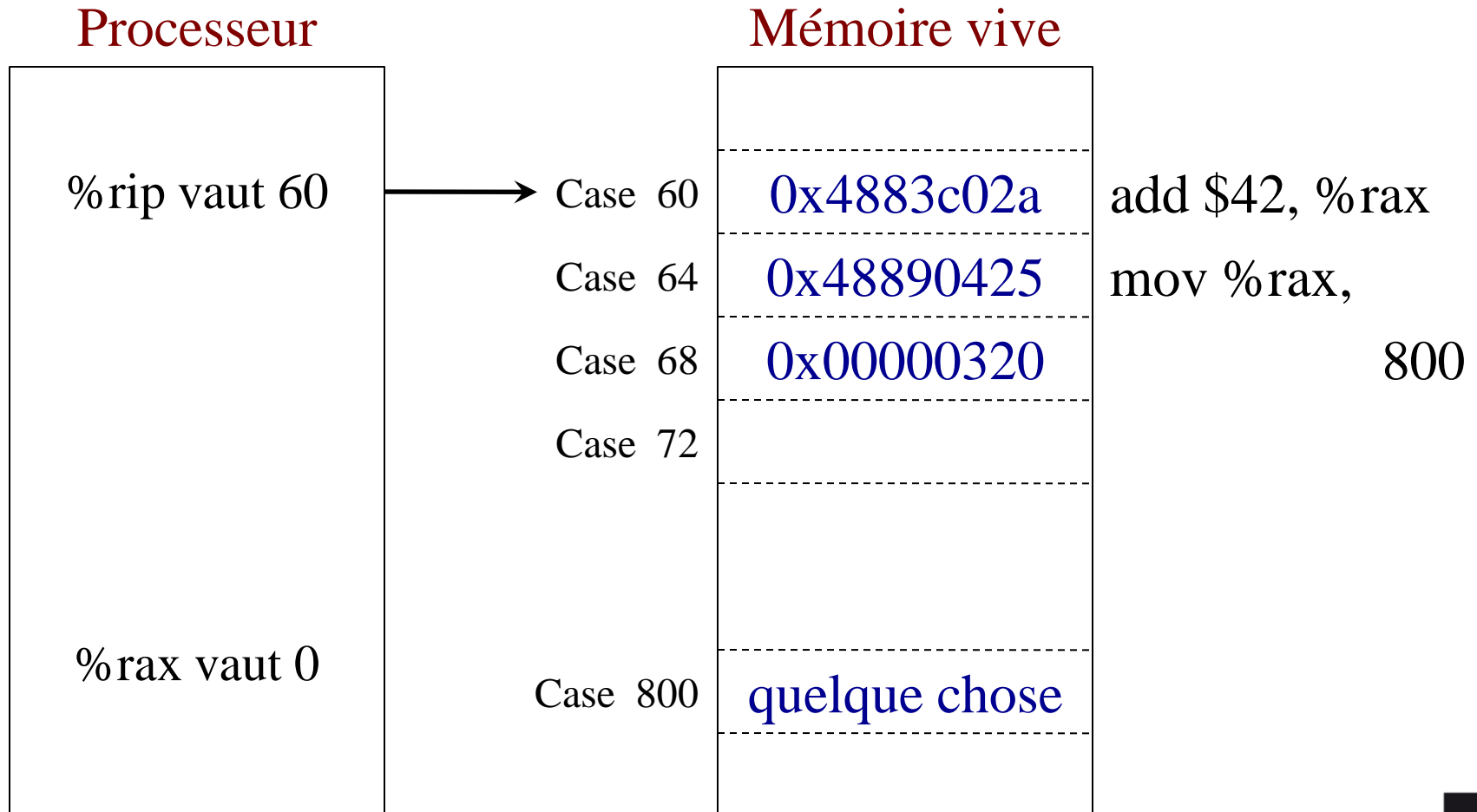
- Les instructions sont stockées dans la mémoire vive
- Le processeur possède un registre indiquant l'instruction courante (registre d'instruction appelé **%rip**)
- Le processeur effectue une boucle qui
  - Charge l'instruction en cours dans un registre du processeur
  - Change l'instruction en cours pour la suivante
  - Exécute l'instruction



# Exécution séquentielle

%rip : registre du processeur indiquant où est l'instruction suivante

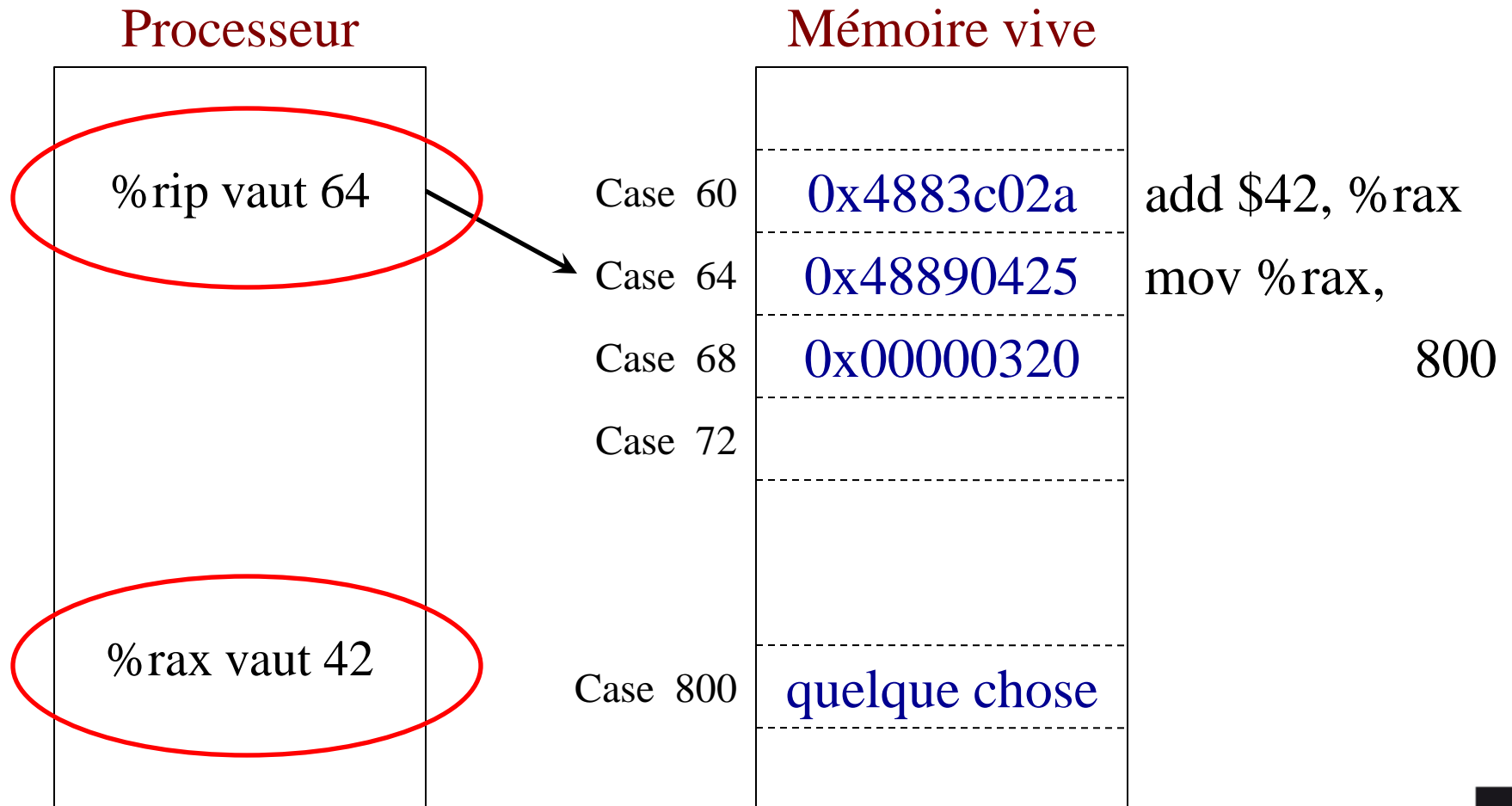
- Un processeur exécute les instructions qui se suivent



# Exécution séquentielle

%rip : registre du processeur indiquant où est l'instruction suivante

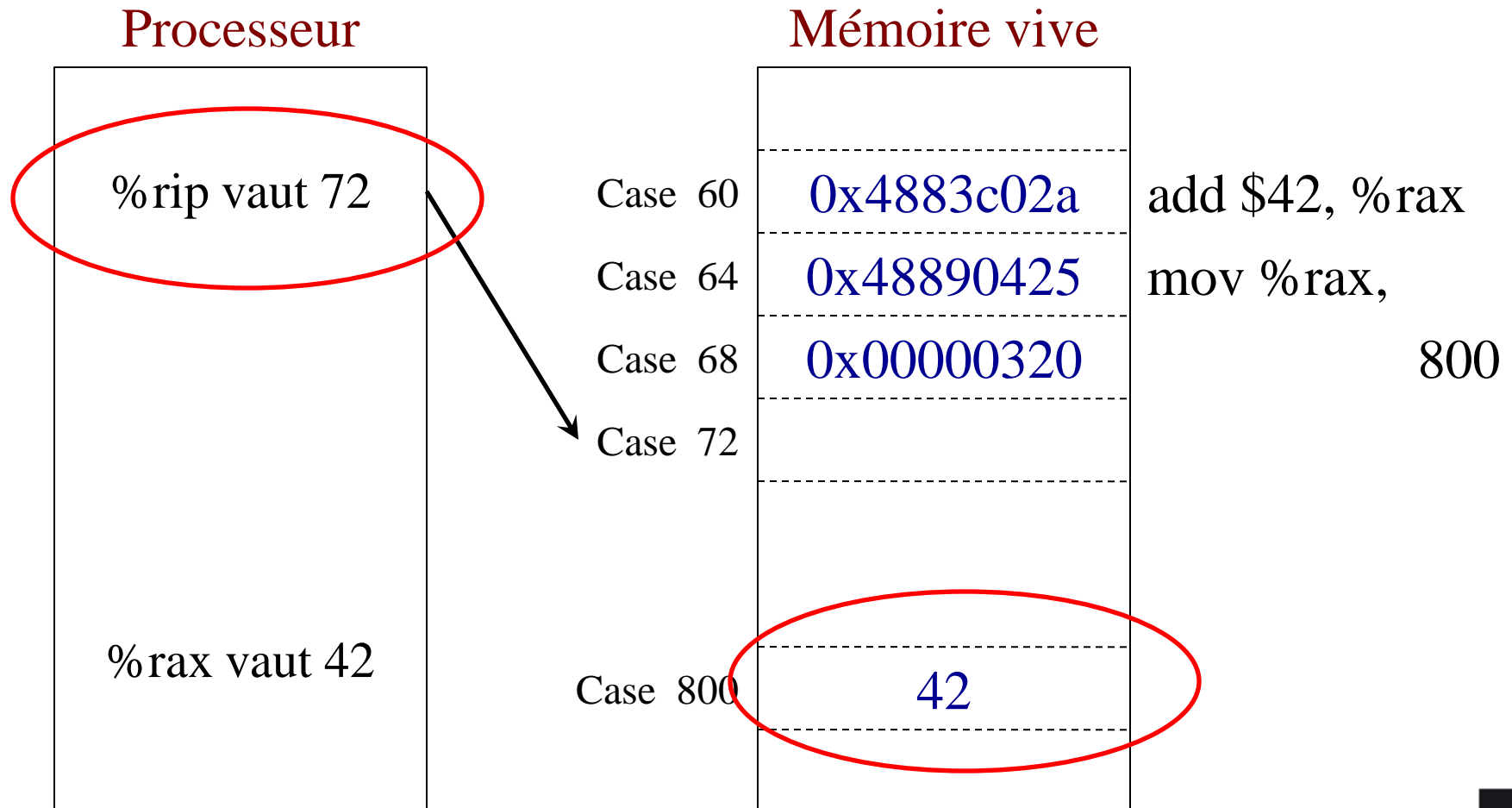
- Un processeur exécute les instructions qui se suivent



# Exécution séquentielle

%rip : registre du processeur indiquant où est l'instruction suivante

- Un processeur exécute les instructions qui se suivent



# Instructions de saut

- Un processeur exécute les instructions qui se suivent  
**Sauf pour les instructions de saut**
- Instruction de saut : instruction qui indique où est le prochain %rip
  - Saut inconditionnel : affecte %rip sans condition
  - Saut conditionnel : affecte %rip sous condition (par exemple si égalité entre deux valeurs)

# Ce qu'il faut retenir

- Une machine est constituée d'un processeur, d'une mémoire vive et de périphériques
- Un processeur exécute de façon séquentielle des instructions qui se trouvent en mémoire
- Chaque instruction est identifiée par un numéro, elle peut
  - Effectuer une opération sur des variables internes (registres)
  - Lire ou écrire en mémoire ses registres
  - Accéder à un périphérique
  - Modifier la prochaine instruction à effectuer (saut)

## II. Logiciels et programmes



# L'ordinateur vu par l'utilisateur

- L'utilisateur installe des **logiciels**

Microsoft office, Chrome, Civilization V...

- Logiciel = ensemble de fichiers

- Fichiers ressources : images, vidéos, musiques...
- Fichiers programmes : ensemble d'opérations et de données destiné à être exécuté par un ordinateur

- In fine, l'utilisateur lance l'exécution de **programmes**

Excel, Word, Chrome, Civilization V, CivBuilder (permet de construire des cartes pour civilization V)...

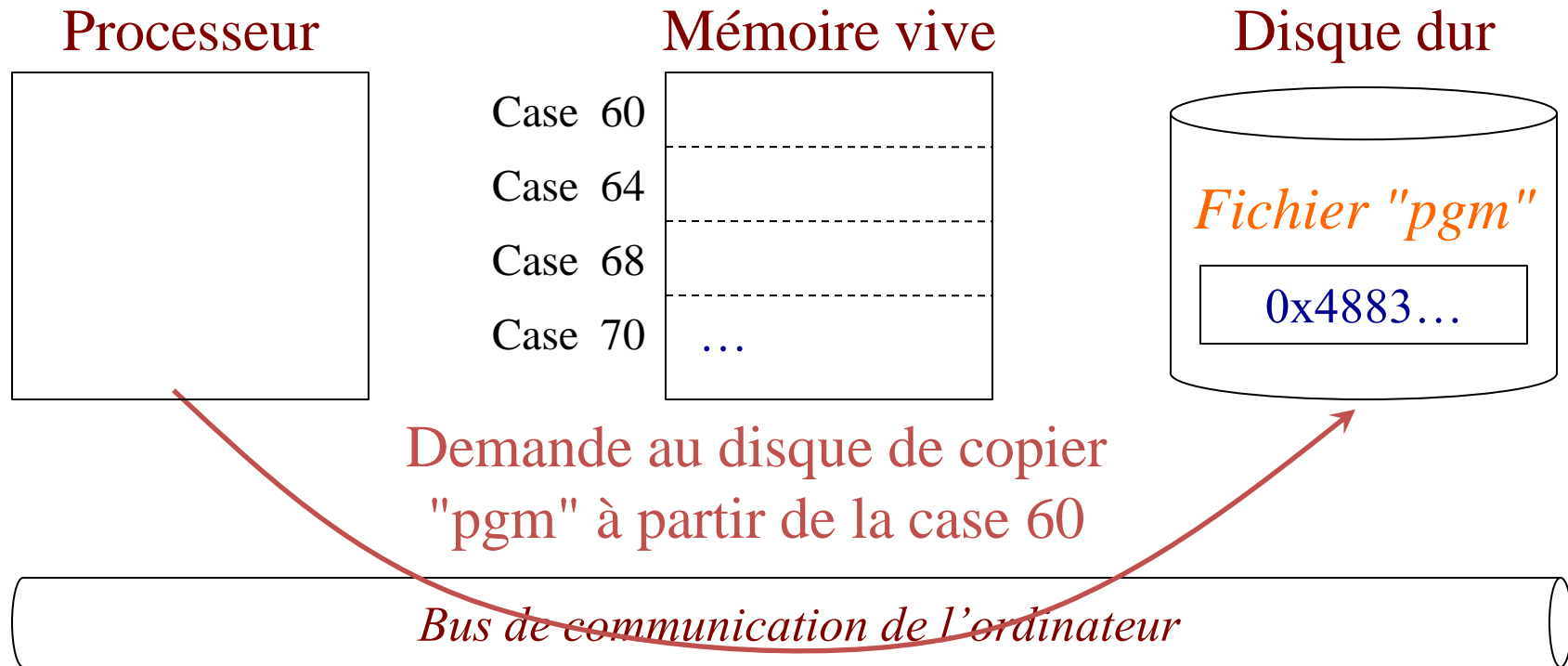


# Qu'est ce qu'un programme

- **Programme binaire** =  
Ensemble d'instructions exécutables par le processeur + des données manipulées par ces instructions
- **Programme source** =  
Ensemble d'opérations abstraites décrivant les actions à effectuer + des données manipulées par ces opérations

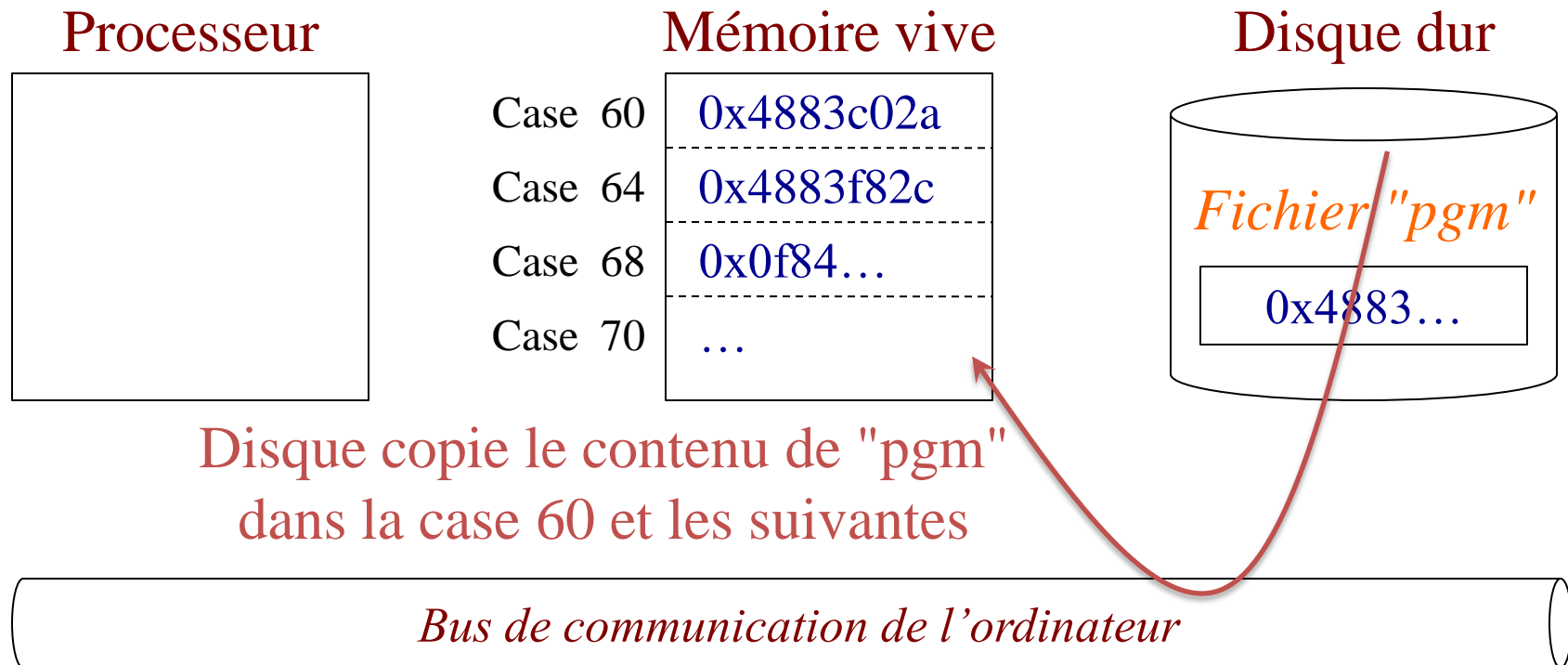
# Exécution d'un programme binaire

- Un binaire doit être chargé en mémoire pour être exécuté  
Chargé à partir du disque dur, du réseau, d'un autre périphérique



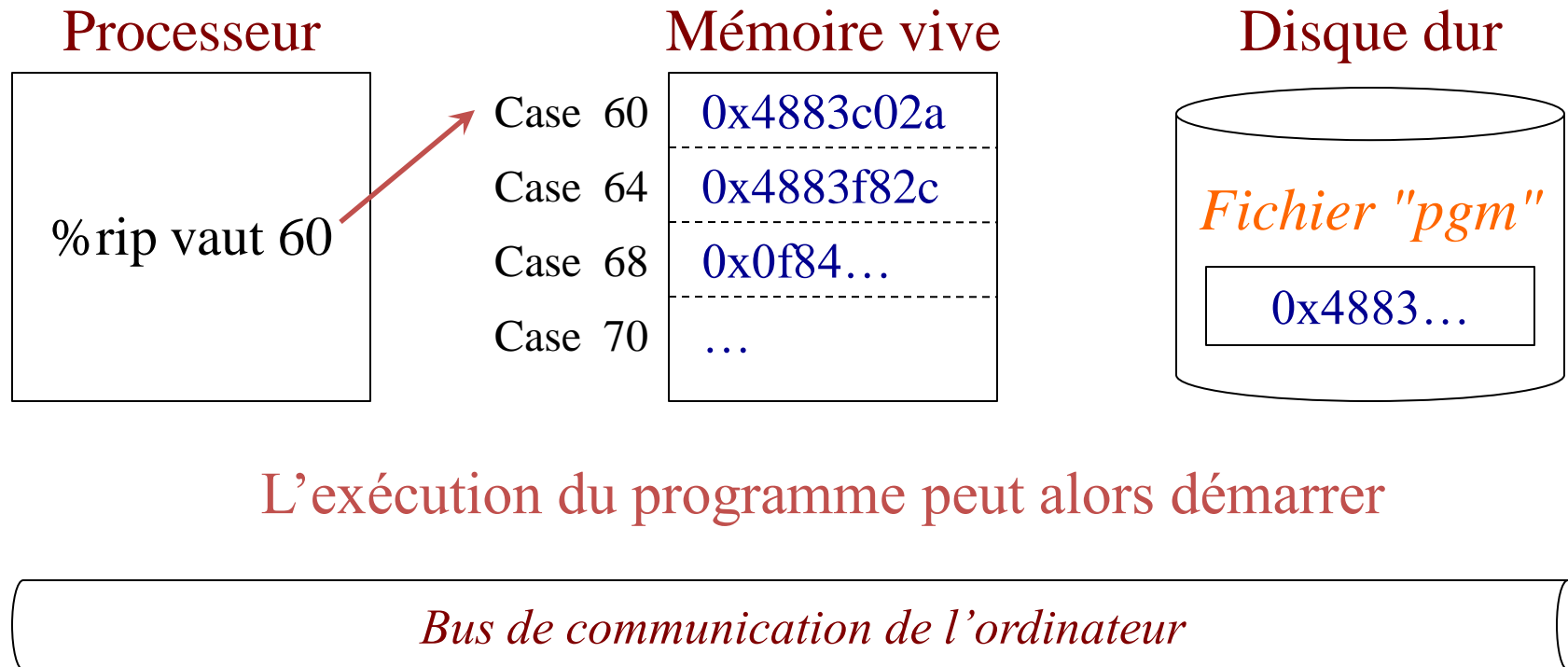
# Exécution d'un programme binaire

- Un binaire doit être chargé en mémoire pour être exécuté  
Chargé à partir du disque dur, du réseau, d'un autre périphérique



# Exécution d'un programme binaire

- Un binaire doit être chargé en mémoire pour être exécuté  
Chargé à partir du disque dur, du réseau, d'un autre périphérique

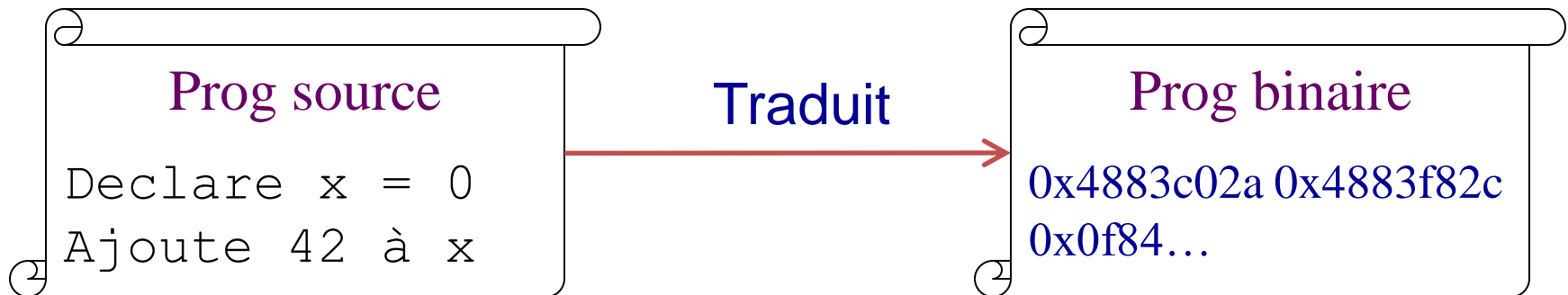


L'exécution du programme peut alors démarrer

*Bus de communication de l'ordinateur*

# Exécution d'un programme source

- Solution 1 : après une traduction vers un programme binaire



En informatique le traducteur s'appelle un compilateur

# Exécution d'un programme source

- Solution 2 : en le faisant interpréter par un autre programme (appelé interpréteur)

## Prog source

```
Declare x = 0  
Ajoute 42 à x
```

Lit et  
interprète



## Interpréteur

1. Lit programme source
2. Pour chaque opération  
    Si Declare ...  
    Si Ajoute ...  
    Si ...

# Quelques exemples de programmes

- Word, Excel ou Chrome sont des programmes binaires
- En général, dans un logiciel de jeux
  - Le jeu lui-même est un programme binaire
  - Capable d'interpréter les mods qui, eux, sont directement des programmes sources (*mod = extension du jeu*)
- Les applications Android sont
  - Interprétées avant Android KitKat (version 4.4)
  - Compilées dès qu'elles sont installées depuis Android KitKat
- Les pages Web interactives sont interprétées



# Processus et système



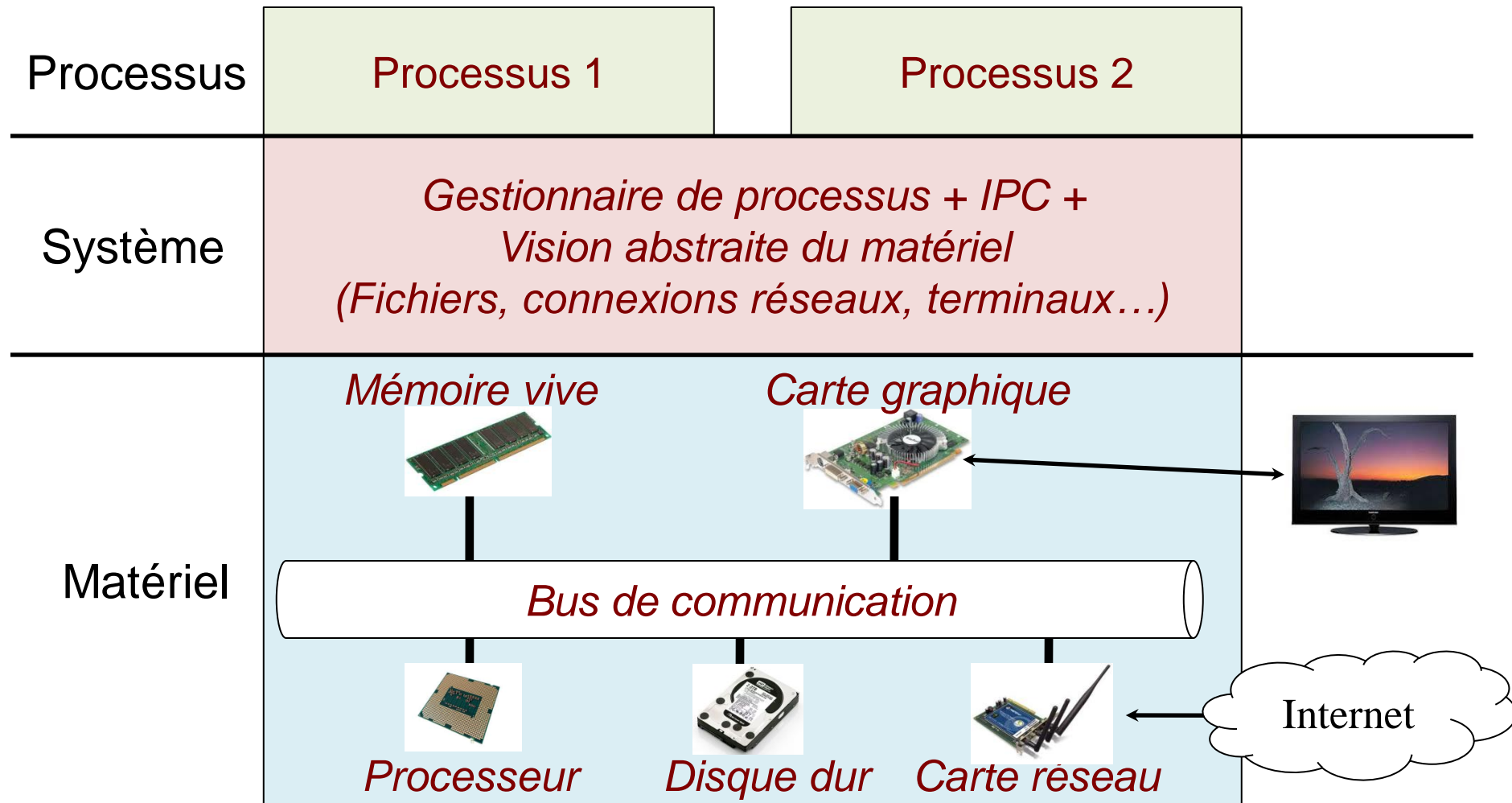
# Du programme au processus

- Un **processus** est un programme en cours d'exécution
  - Contient bien sûr les opérations du programme
  - Mais aussi son état à **un instant donné**
    - Données en mémoire manipulées par le programme
    - État des registres du processeur
    - État des périphériques (fichiers ouverts, connexions réseaux...)

# Gestion des processus

- Le **système** est un logiciel particulier qui gère les processus  
(Le système est le seul programme qu'on n'appelle pas processus quand il s'exécute...)
  
- Rôle du système
  - Démarrer un processus  
(en chargeant le binaire ou l'interpréteur adéquat)
  - Arrêter un processus
  - Offrir une vision abstraite du matériel aux processus
  - Offrir des mécanismes de communication inter-processus (IPC)

# Architecture globale à l'exécution



# Naissance des premiers systèmes UNIX

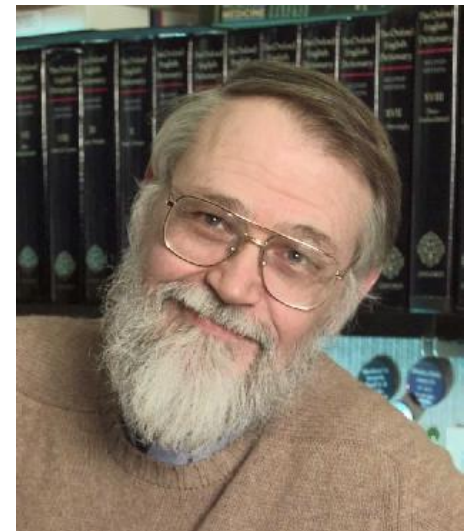
- 1969 : première version d'UNIX en assembleur
- 1970 : le nom UNIX est créé
- 1971 : invention du C pour réécrire UNIX dans un langage de haut niveau



Ken Thompson



Dennis Ritchie



Brian Kernighan

# Objectif du module

- Étude des systèmes Unix par l'exemple
- À l'aide du langage bash (CI1 et 4)
  - Langage interprété par le programme bash
  - Langage spécialisé dans la gestion de processus
- Comprendre
  - La notion de fichier (CI2 et 3)
  - La notion de processus (CI5)
  - Des mécanismes de communication inter-processus (CI6 à 8)
  - La notion de virtualisation (CI9)

# Notions clés du cours

- Un ordinateur
  - Est composé de : mémoire, processeur, périphérique et bus
  - Un processeur exécute des instructions se trouvant en mémoire
- Un logiciel contient des fichiers
  - Ressources (images, sons, textures...)
  - Programmes (source et/ou binaire)
- Un programme est une suite d'opérations + des données
- Un processus est un programme en cours d'exécution
  - Opérations + état à un instant donné
- Le système gère les processus et abstrait le matériel

# Annexe : représentation des données



# Représentation des entiers

- Les octets sont regroupés pour former des valeurs entières

*(souvent par 1, 2, 4 ou 8 octets)*

- Peut être vu comme un naturel (dans N)

$$0xe159 = 14 \cdot 16^3 + 1 \cdot 16^2 + 5 \cdot 16 + 9$$

⇒ 0x159 représente 57689

- Ou comme un relatif (dans Z)

$$0xe159 + 0x1ea7 = 0x0000 + \text{une retenue}$$

0x1ea7 est donc l'inverse de 0xe159

$$0x1ea7 = 1 \cdot 16^3 + 14 \cdot 16^2 + 10 \cdot 16 + 7 = 7847$$

⇒ 0xe159 représente -7847

Le nombre  
0xe159

0xe1
0x59
0x61
0xf0

# Représentation des valeurs de vérités

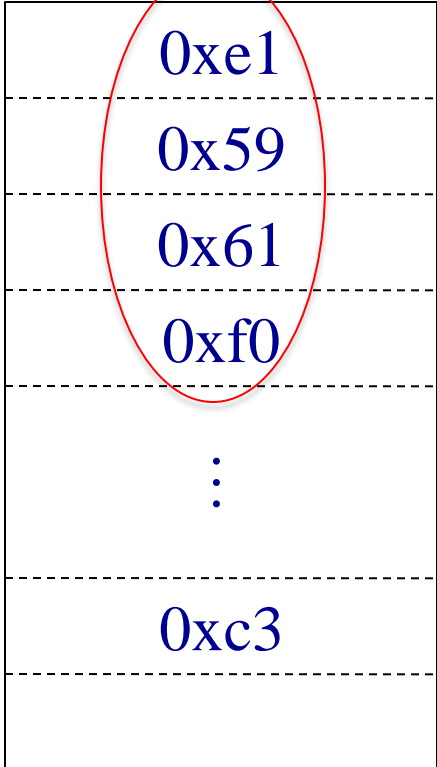
□ Booléen : valeur pouvant valoir vrai ou faux

Une valeur vrai

□ Peut être stocké sur 1 bit, 1 octet, 2 octets, 4 octets, 8 octets...

□ Convention :

- 0 vaut faux
- Toute autre valeur vaut vrai



0xe1
0x59
0x61
0xf0
⋮
0xc3

# Représentation des caractères

- Un octet peut être vu comme un caractère
- Table ascii pour faire la correspondance

0x1f	↵		0x41	A		0x61	a
0x20	' '		0x42	B		0x62	b
...	...		0x43	C		0x63	c
0x30	0		0x44	D		0x64	d
0x31	1		0x45	E			e
0x32	2		0x46	F		0x66	f
0x33	3		0x47	G		0x67	g
...	...		...	...		...	...

*Ceci est un 'a'*

0xe1
0x59
0x61
0xf0
⋮
0xc3

