- Introduction
- Microcontrôleur / Microprocesseur
  - Terminologie
  - Comparaison de solutions
  - Choix d'une architecture
- Apports d'un système d'exploitation
  - Définition d'un OS
  - Abstraction des périphériques
  - Exécution des tâches
  - Mémoire virtuelle et MMU
- Linux pour l'embarqué
  - Pourquoi linux ?
  - Composants d'un système linux
  - Démarrage du système
  - Temps réel
  - Les principaux systèmes d'exploitation dans l'embarqué
  - Construire son système
  - Amélioration des performances du noyau linux

Qu'est ce qu'un système embarqué ?

Un système embarqué est un ensemble électronique et/ou informatique intégré comme composant d'un environnement plus important.

- Un système embarqué se définit surtout par les contraintes auxquelles il est soumis.
- L'identification précise des contraintes doit se faire dès la conception du système.

- **Embedded** : enfoncer, sceller, noyer, enchâsser, incruster

=> Système enfoui ou incorporé

Contraintes sur un système embarqué

Contraintes matérielles

- Performance
- Encombrement
- Autonomie

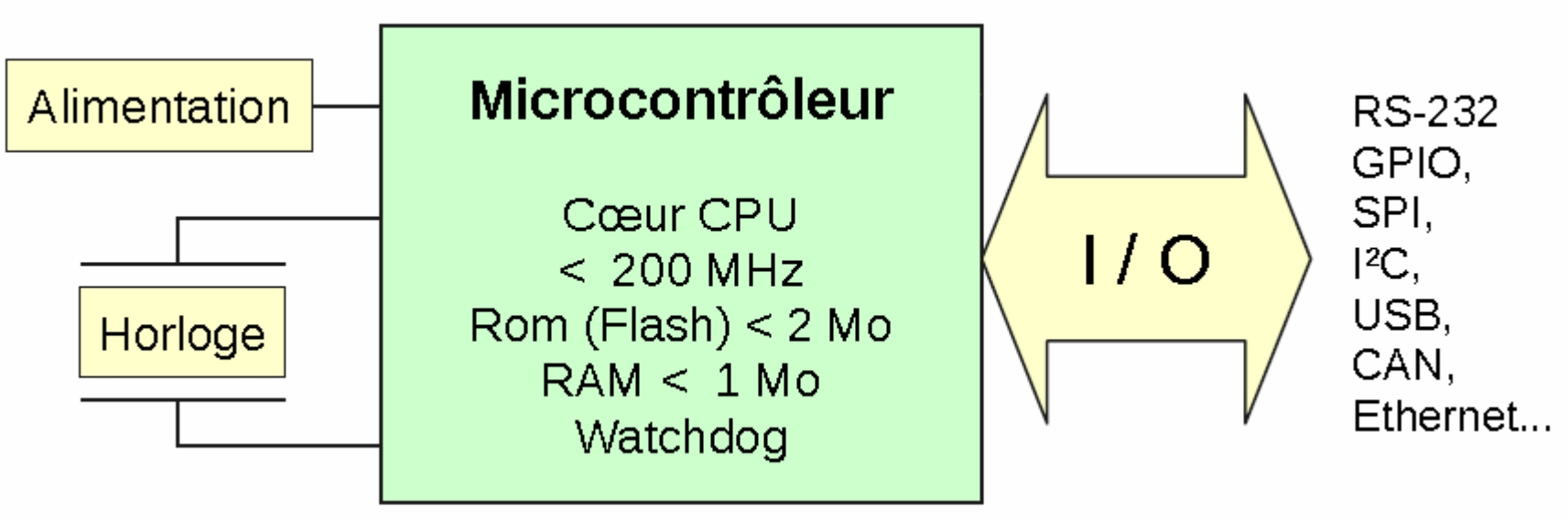
Contraintes logicielles

- Performance
- Robustesse
- Sécurité

Contraintes économiques

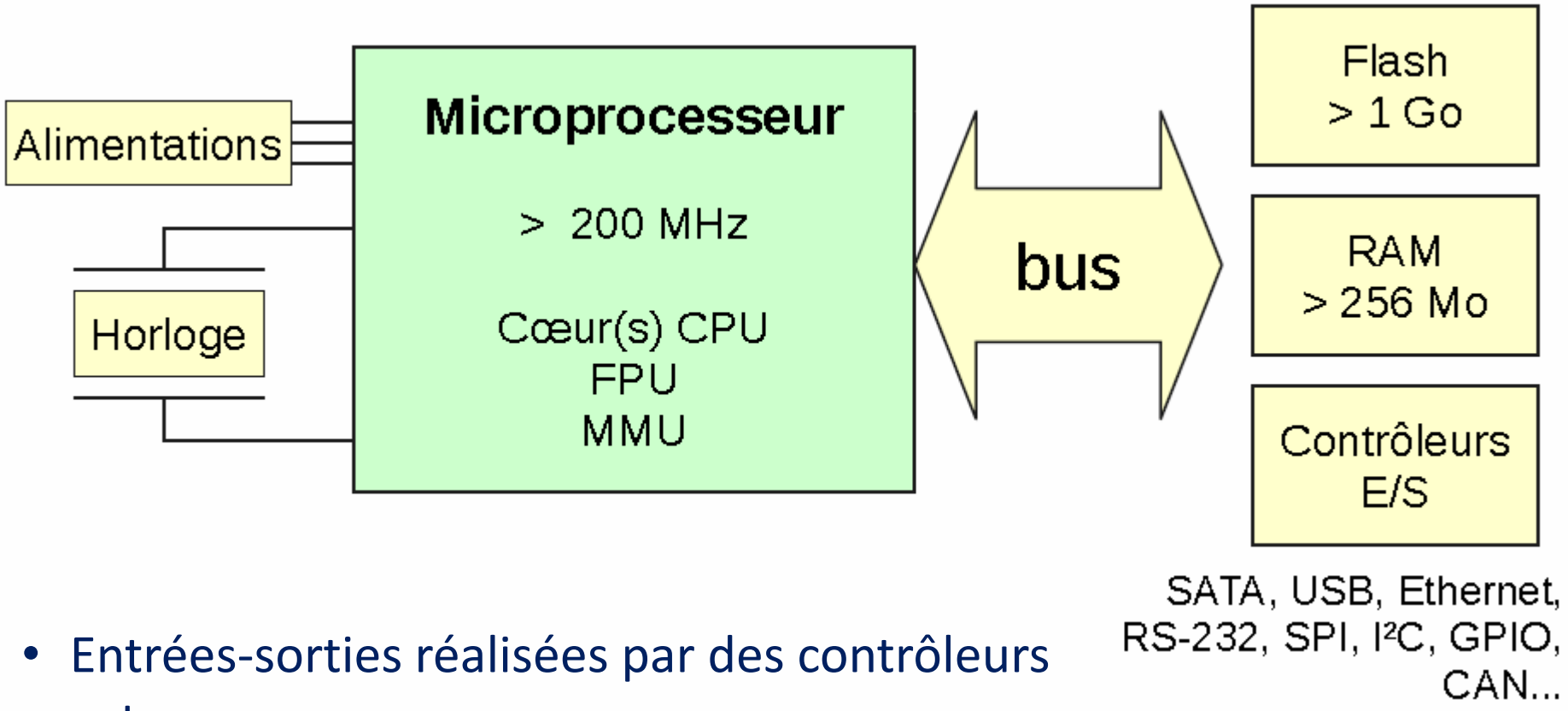
- Concurrence
- Coûts
- Évolutivité

Introduction - terminologie



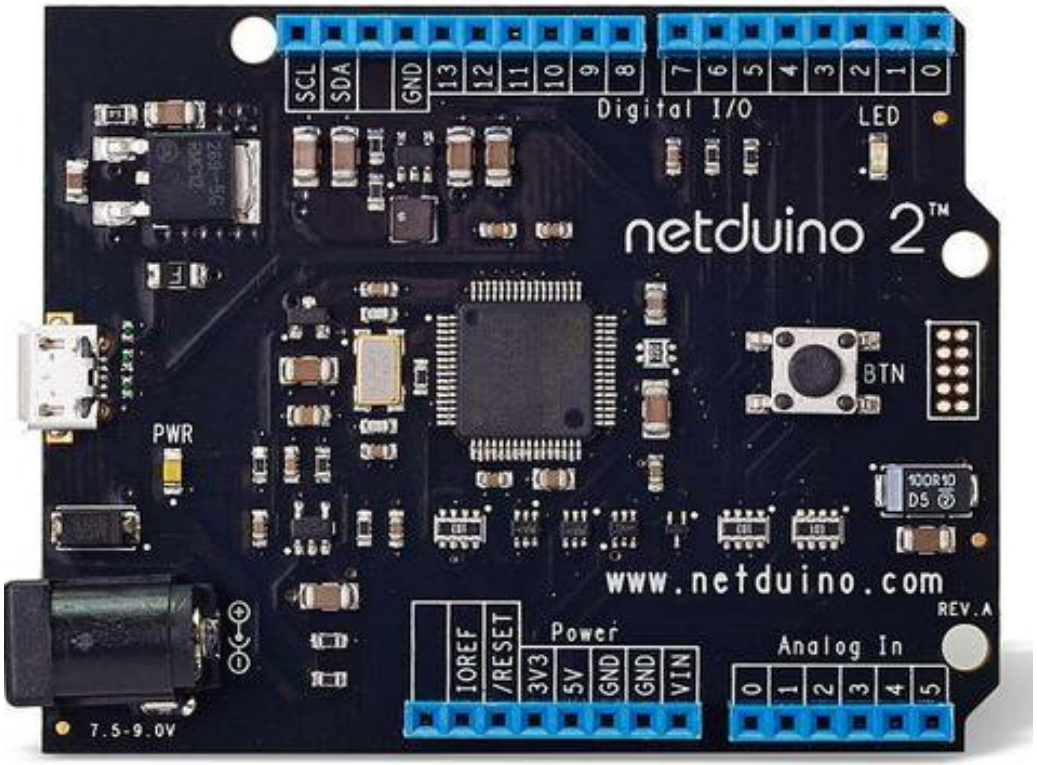
- Mise en œuvre électronique simple.
- Déterminisme et fiabilité de fonctionnement.
- Généralement pas de système d'exploitation (ou minimal).
- Nombreux fabricants :
  - Atmel, Freescale, Hitachi, Intel, Microchip, STMicro, TI, ...

Introduction - terminologie

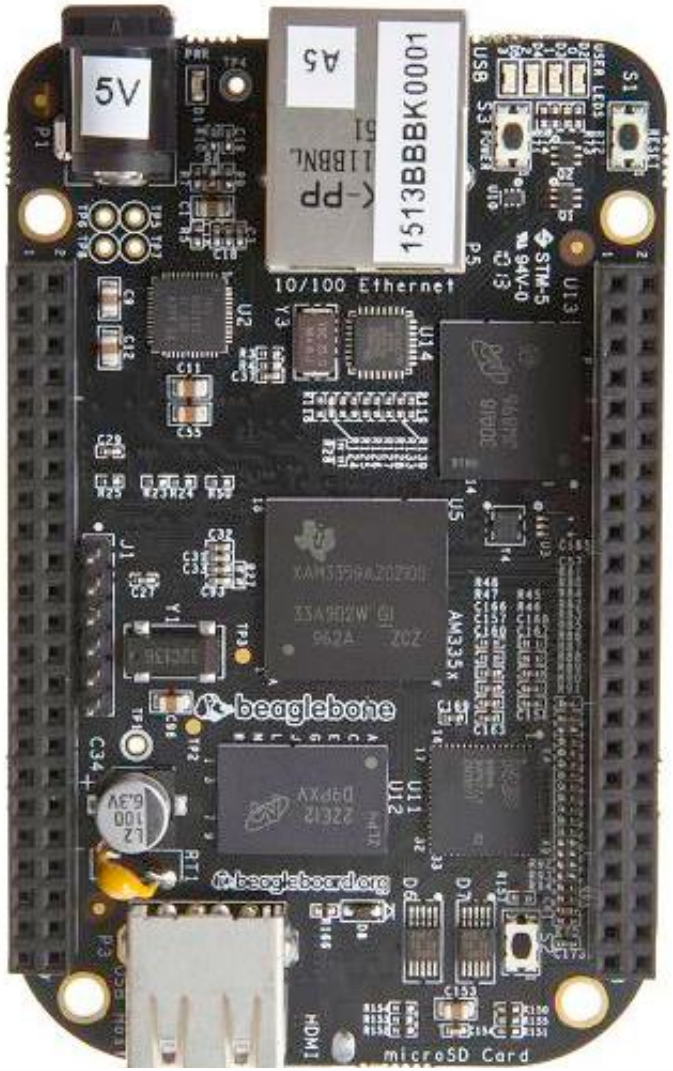


- Entrées-sorties réalisées par des contrôleurs externes au processeur
- Mise en œuvre électronique beaucoup plus complexe
- Optimisé pour l'utilisation d'un système d'exploitation
- Quelques familles : Arm, x86, M68k, PowerPC

Comparaison des solutions

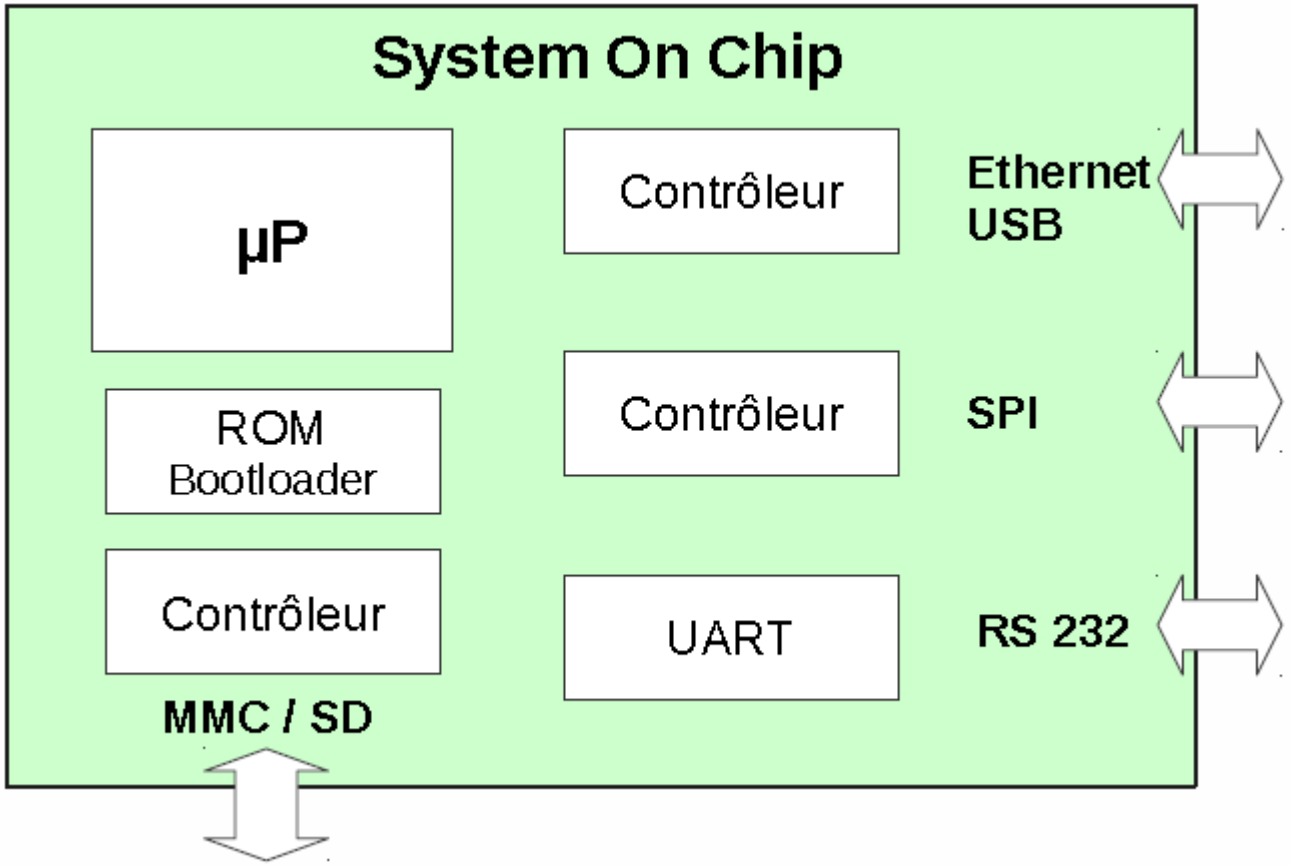


Système à micro-contrôleur
(STM32F2)



Système à micro-processeur
(T.I. AM3359 de BeagleBone Black)

Introduction - terminologie



- Contrôleurs d'entrées-sorties déjà incorporés
- Intégration électronique encore assez complexe
- Souvent peu d'entrées-sorties industrielles et analogiques

Introduction - terminologie

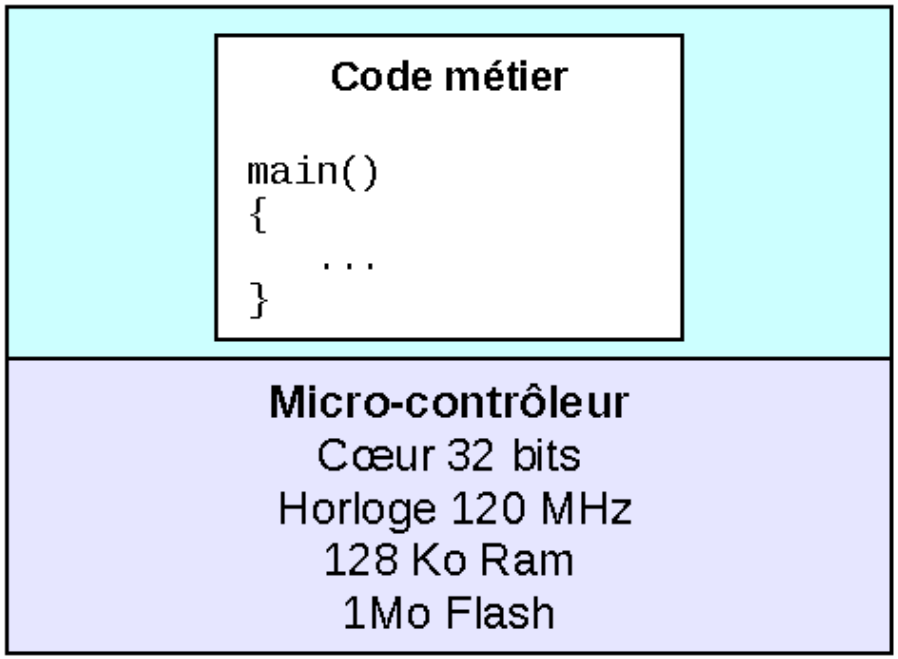
BCM2836 Broadcom

- CPU : ARM Cortex A7
- FPU : VFPv3
- GPU : VideoCore IV
- 1Go Memory
- Support HDMI transmitter

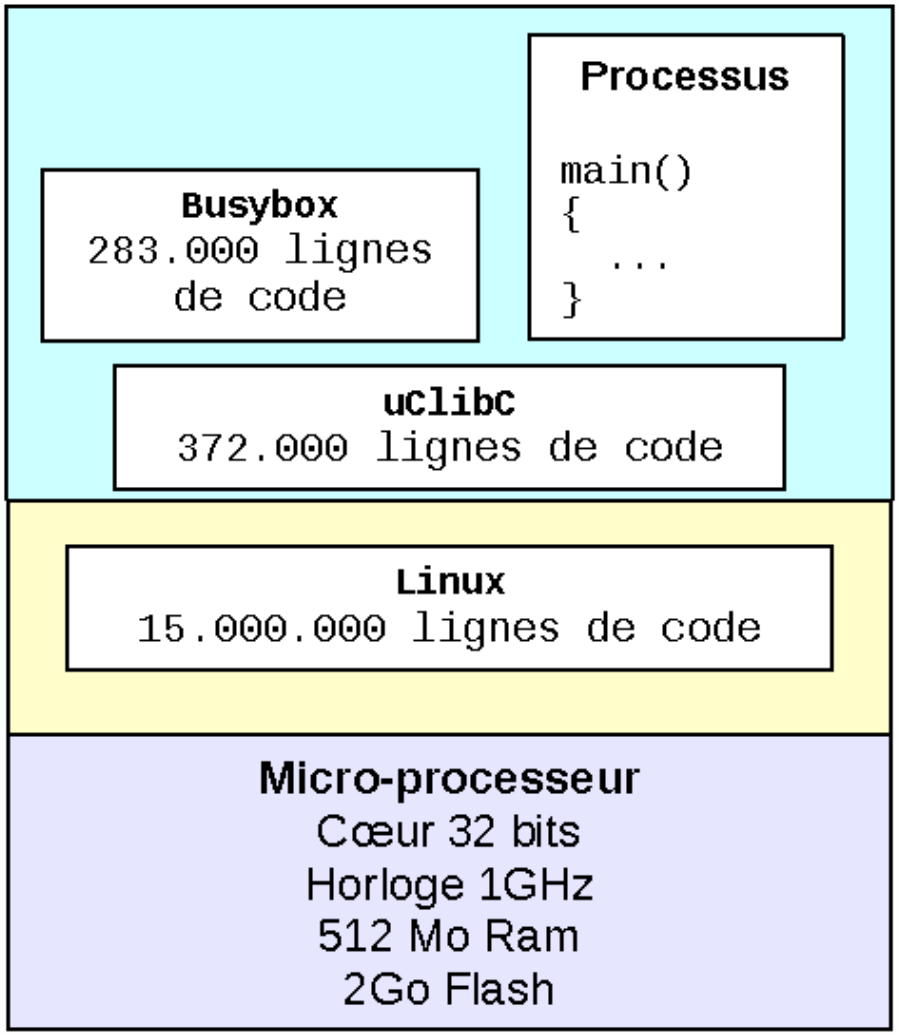
A13 Allwinner

- CPU : ARM Cortex A8
- FPU : VFPv3
- GPU : Mali400
- VPU : CedarX (DSP)
- Support HDMI transmitter

Comparaison des solutions



Système à micro-contrôleur
(STM32F2)



Système à micro-processeur
(T.I. AM3359 de BeagleBone Black)

Choix d'une architecture matérielle

Microcontrôleur

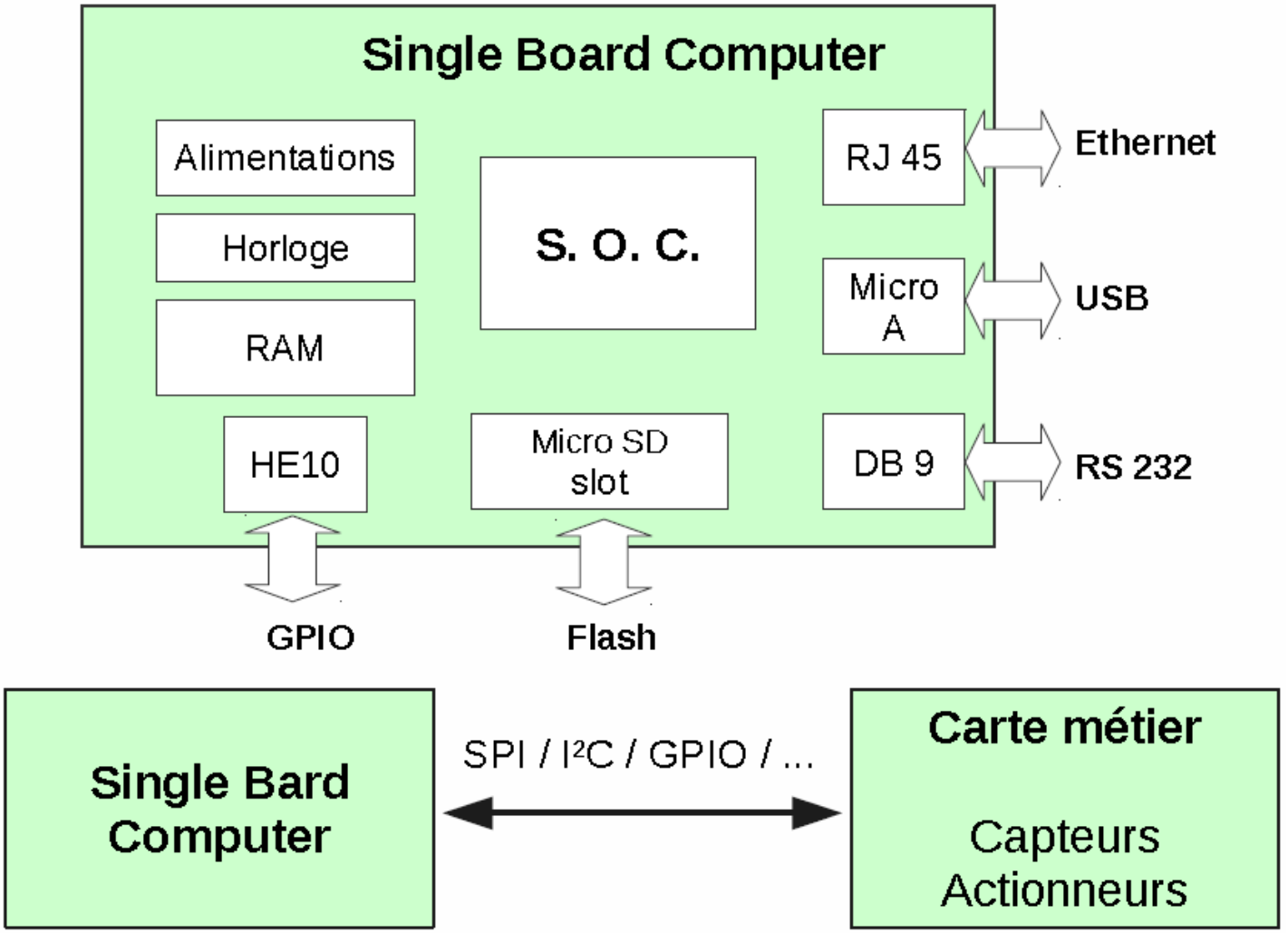
- **Prix** : conception et réalisation PCB, cout unitaire.
- **Simplicité** : fiabilité, code certifiable.
- **Prédictibilité** : temps d'exécution, déterminisme.

Microprocesseur / SOC (système d'exploitation)

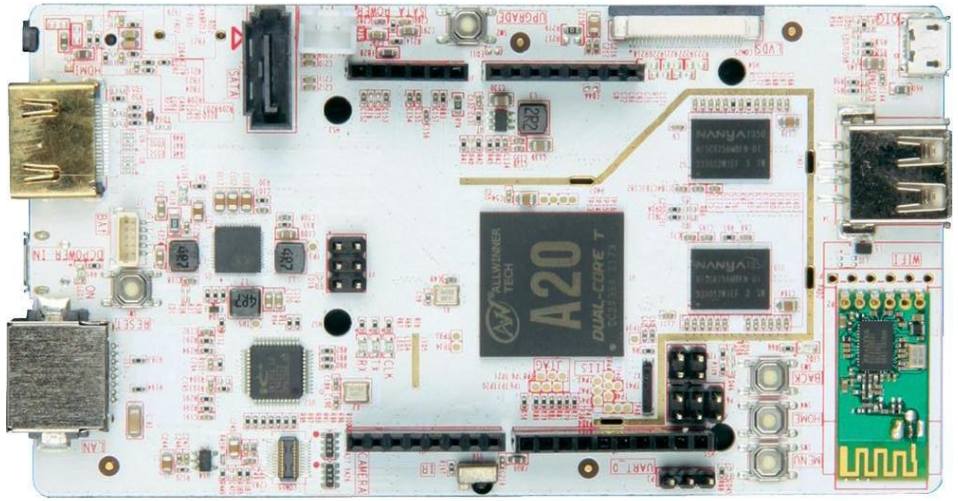
- **Puissance** : calcul, mémoire, optimisation.
- **Evolutivité** : isolation du code métier par rapport au matériel, portabilité.
- **Richesse applicative** : piles de protocoles, services…

Choix d'une architecture matérielle

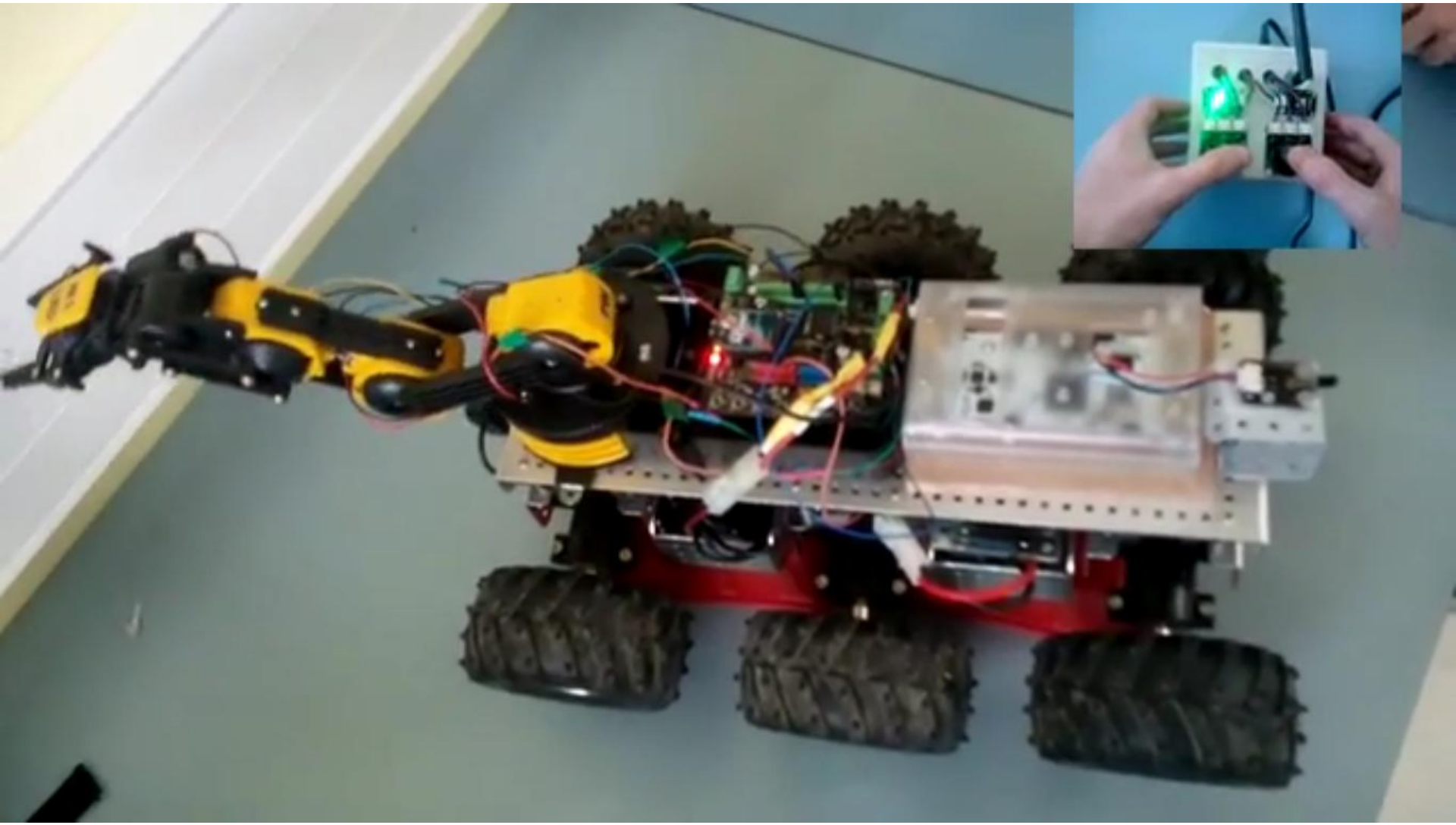
Prototypage / projet personnel



Choix d'une architecture matérielle
Prototypage / projet personnel

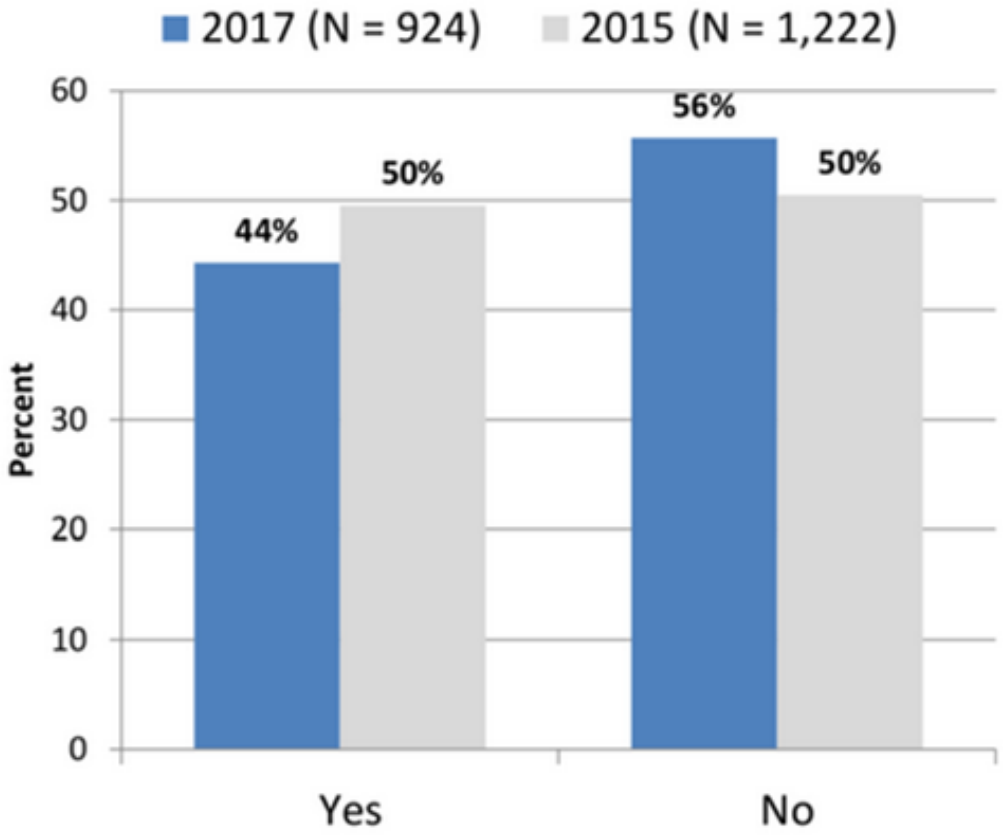


Choix d'une architecture matérielle
Prototypage / projet personnel



Choix d'une architecture matérielle

Prototypage / projet personnel

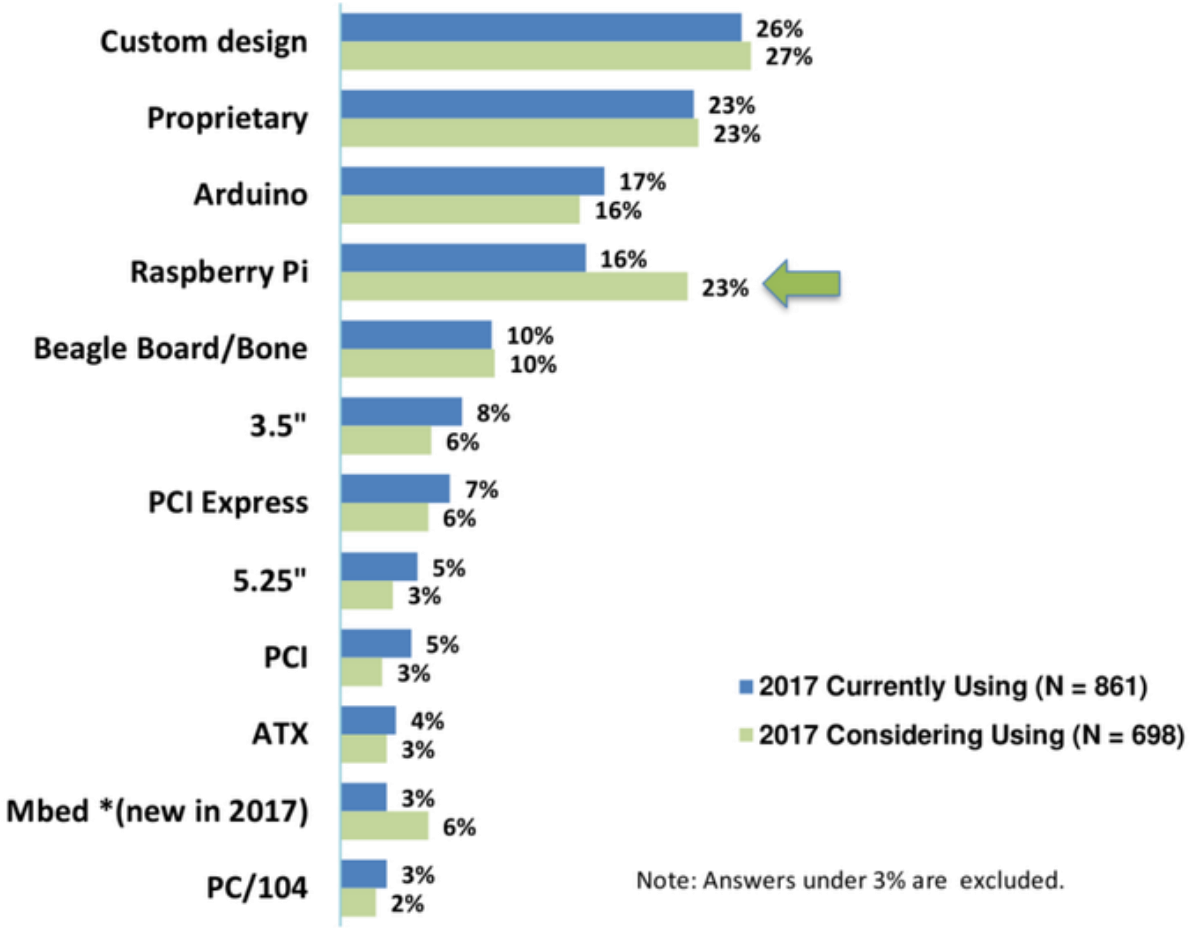


Did you start your current embedded design with a development board?

https://www.cnx-software.com/2017/08/15/aspencore-2017-embedded-markets-study-programming-languages-operating-systems-mcu-vendors-and-more/

Choix d'une architecture matérielle

Prototypage / projet personnel



Which form factor boards are you currently using, and considering using ?

https://www.cnx-software.com/2017/08/15/aspencore-2017-embedded-markets-study-programming-languages-operating-systems-mcu-vendors-and-more/

Choix d'une architecture matérielle

Petite série – Startup

System on Module

Carte porteuse

Alimentation
Capteurs
Actionneurs

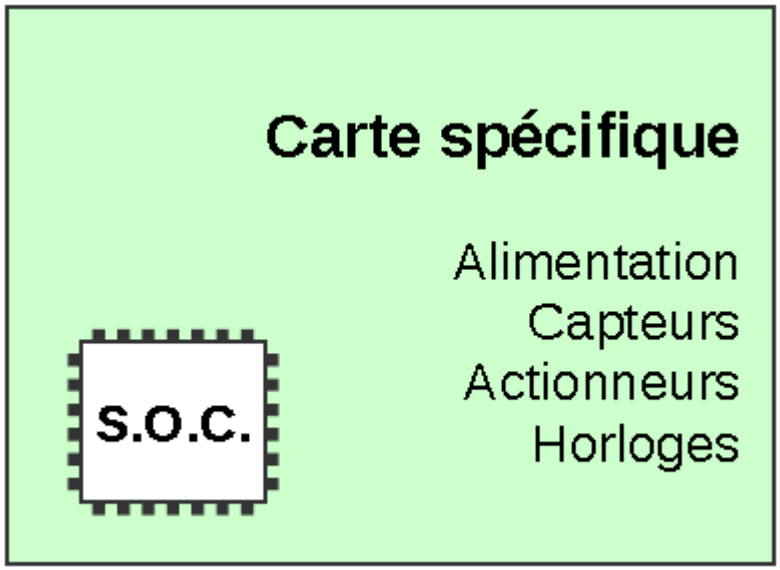
ARIA G25 - LINUX EMBEDDED MODULE

PC on a stick : Système complet tenant sur une carte
Approchant la taille d'une clé USB

By Pe wiki editor (Own work) [CC BY-SA 3.0], via Wikimedia Commons

Choix d'une architecture matérielle

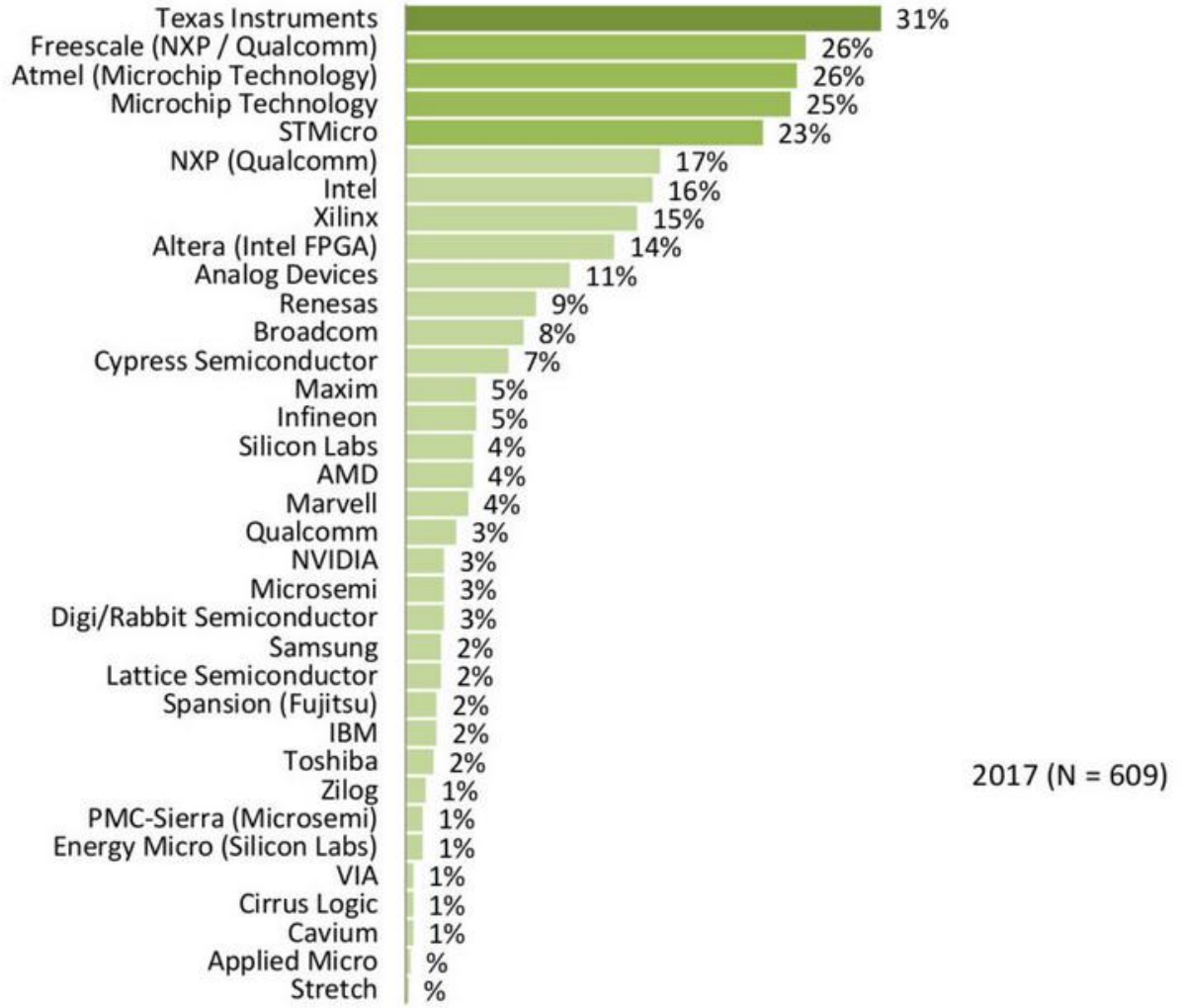
Grande série – production industrielle



- Rarement intéressant en dessous d'une dizaine de milliers d'unité.
- Coûts importants de design, routage, banc de test, validation, etc.
- Frais de production avantageux.
- Externalisation de la conception : attention à la propriété intellectuelle

Choix d'une architecture matérielle

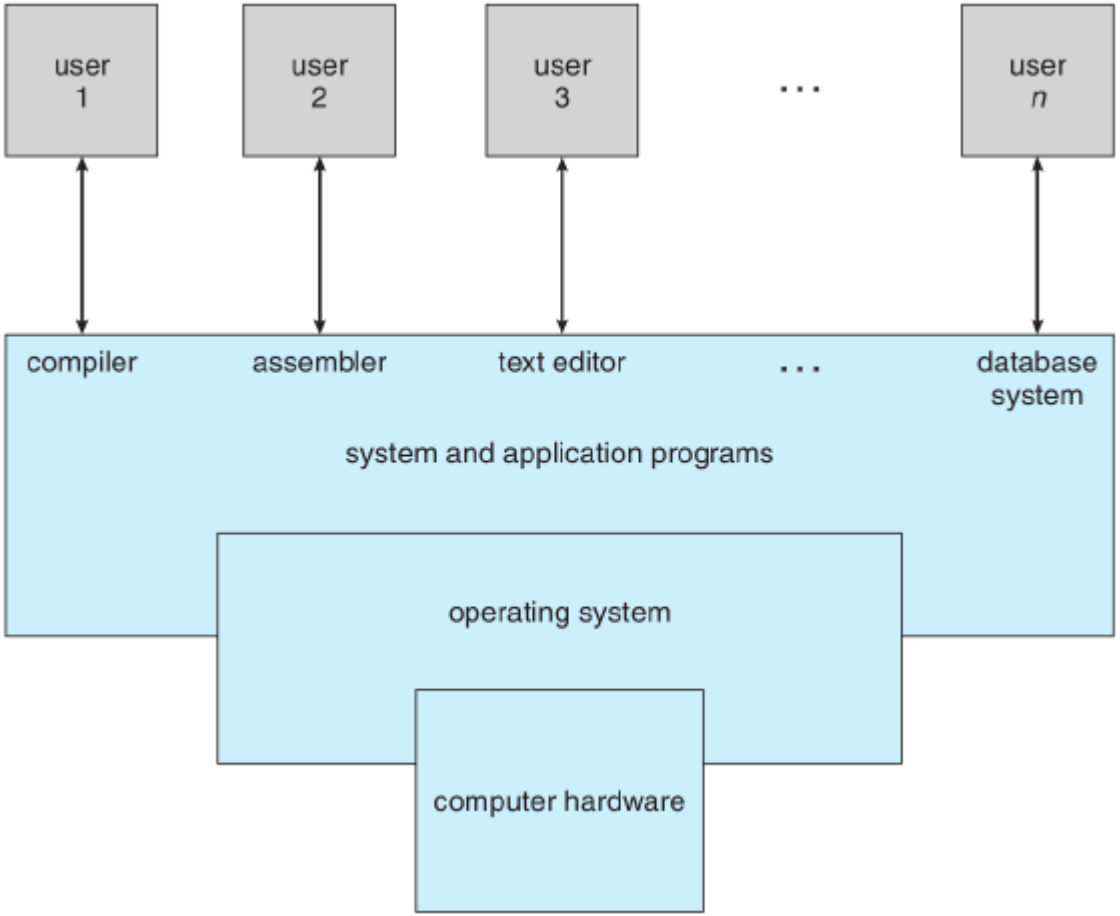
Grande série – production industrielle



Définition d'un OS

Logiciel qui gère le matériel et fournit un environnement pour les programmes applicatifs en exécution

Pour l'utilisateur :
interface fournie avec l'ordinateur, le smartphone...

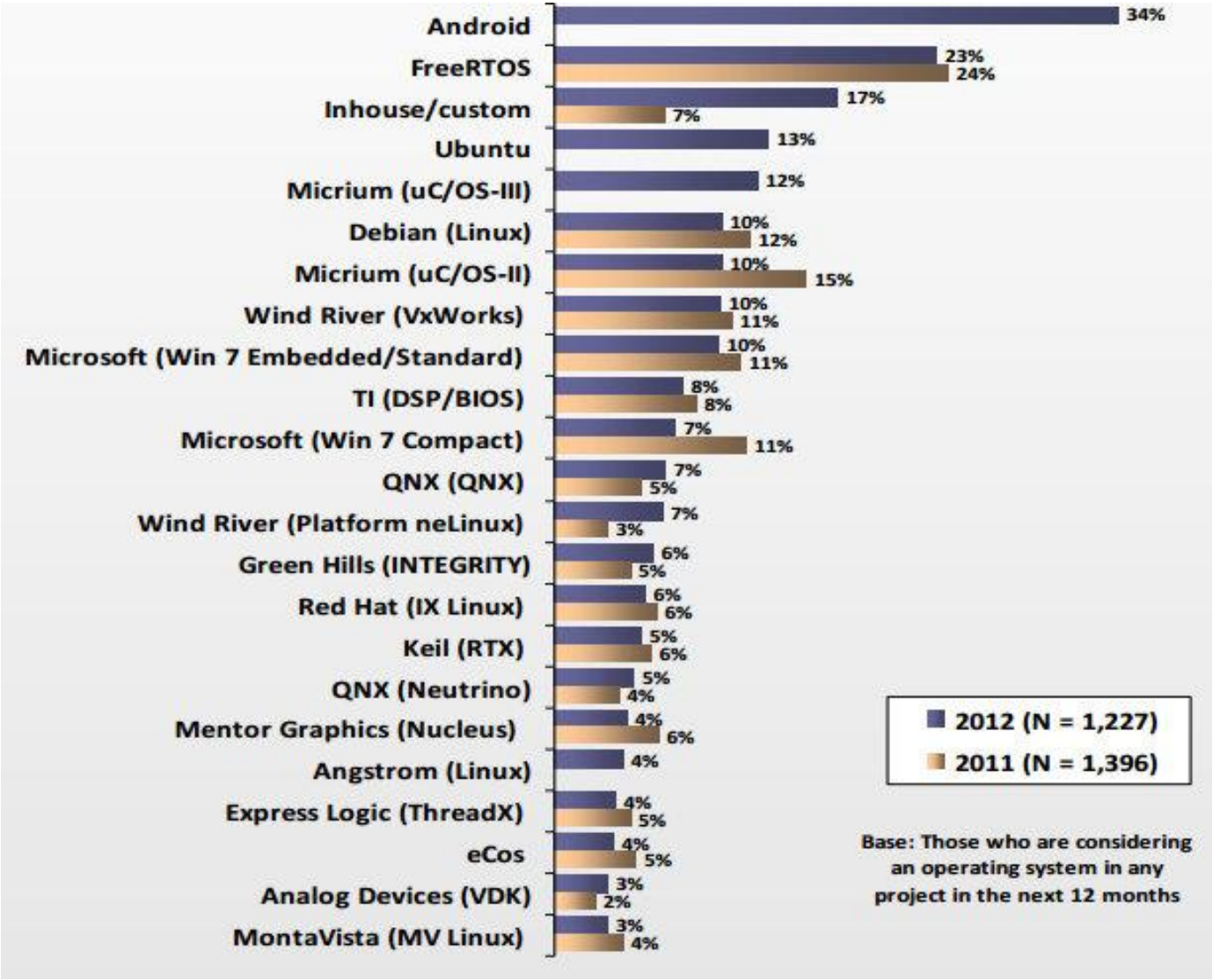


Utilisation des OS

Category	Source	Date	Linux	Unix and Unix-like (not incl. Linux)	Windows	In-house	Other
Desktop, laptop	Net Applications[215]	September 2018	2.21% (excl. Chrome OS) plus 0.29% ChromeOS	9.52% (macOS)	87.56% (all versions)		0.37%
Smartphone, tablet	StatCounter Global Stats[216]	September 2018	73.19% (Android)	24.26% (iOS)	0.36%		2.19%
Server (web)	W3Techs[217]	Apr 2017	66.6–37% (of the known-for-sure lower bound of Linux share: Ubuntu 35.8%, Debian 31.9%, CentOS 20.6%, Red Hat (RHEL) 3.3%, Gentoo 2.7%, Fedora 0.9%)	c. 1% (BSD; Unix-like could be up to 30.18%, then "Unknown" needs to be known to be not Linux) 66.6% Unix-like share is mostly Linux; the "Unknown" part, there-of 43.1%, is assumed to be also Linux (for upper bound of that column), could be some non-Linux or e.g. any of the named Linux distributions in the Linux column.	33.5% (Windows Server 2016, W2K12, W2K8)		
Supercomputer	TOP500	Nov 2017	100% (Custom)				
Mainframe	Gartner[209]	Dec 2008	28% (SLES, RHEL)	72% (z/OS) UNIX System Services			
Video game consoles	VGChartz[218]	Jan 2018		35.04% (PS4, PS3, Vita, PSP)	16.63% (Xbox One, Xbox 360)	48.32% (Switch, Wii U, Wii, 3DS, DS)	
Embedded	UBM Electronics[219]	Mar 2012	29.44% (Android plus other non-Android Linux)	4.29% (QNX)	11.65% (WCE 7)	13.5% ("Inhouse/custom" is most popular, single choice)	41.1%

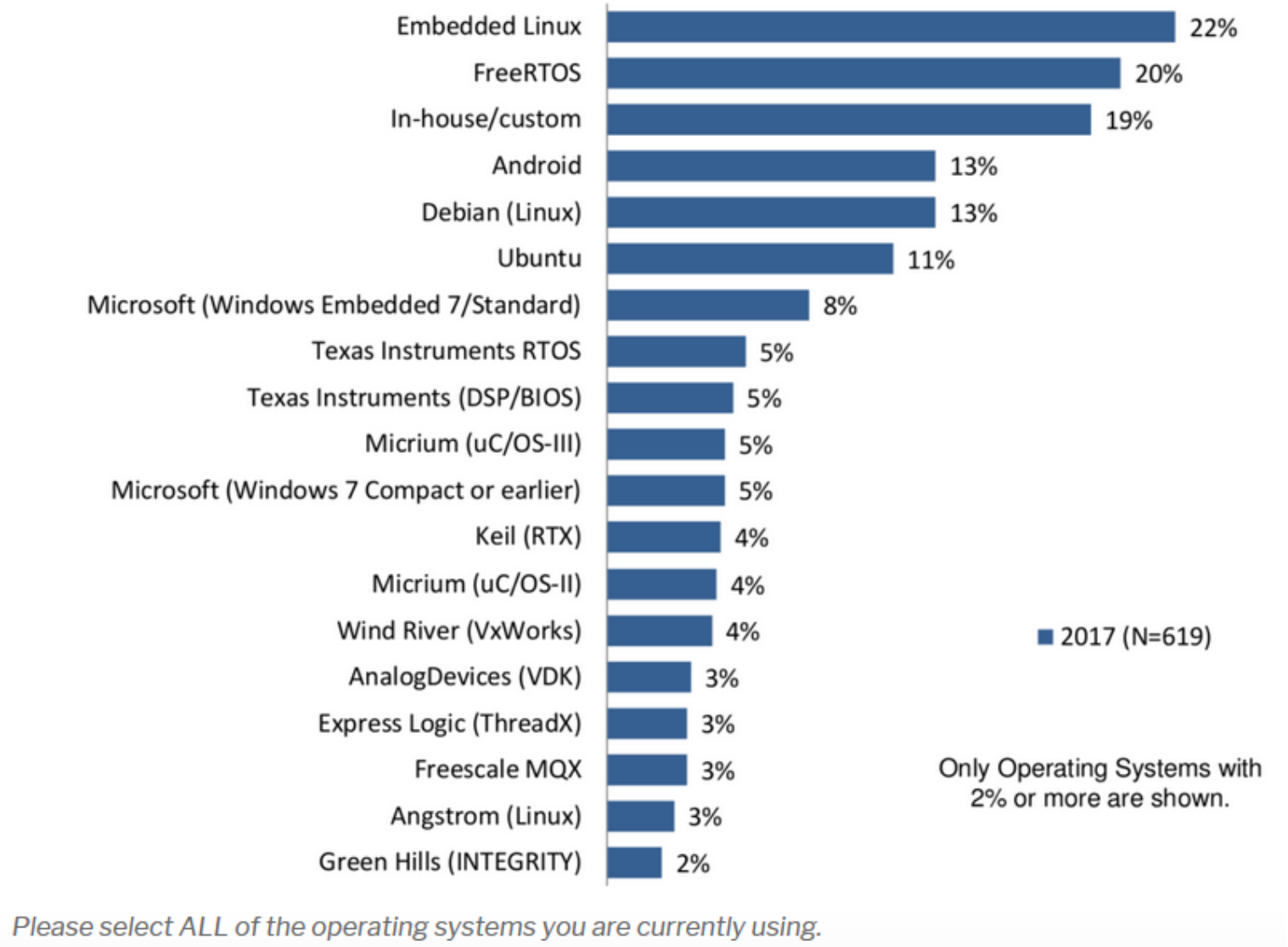
Utilisation des systèmes d'exploitation par catégories
(https://en.wikipedia.org/wiki/Usage_share_of_operating_systems)

Utilisation des OS



https://www.cnx-software.com/2012/11/28/2012-embedded-market-study-software-development-processors/

Utilisation des OS



https://www.cnx-software.com/2017/08/15/aspencore-2017-embedded-markets-study-programming-languages-operating-systems-mcu-vendors-and-more/

Les principaux systèmes d'exploitation dans l'embarqué – Non linux

VxWorks

- Fut le noyau temps réel le + utilisé dans l'industrie
- Développé par Wind River (acquis par Intel en 2009)
- Support en natif de TCP/IP et interface Posix
- Très utilisé par les systèmes embarqués contraints
- Peu répandu dans les systèmes grand public (peu adapté au multimédia)
- Virage vers le logiciel libre : Wind River Linux

QNX

- Développé par QNX Software
- Noyau temps réel de type Unix, conforme à Posix
- Intègre l'environnement graphique Photon (proche de X Window System)
- Mise à disposition de la majorité des outils GNU
- Peut être utilisé gratuitement pour des applications non commerciales et l'éducation
- Très faible empreinte mémoire

Les principaux systèmes d'exploitation dans l'embarqué – Non linux

µC/OS et µC/OS II

- Destiné à des environnements de très petite taille comme des microcontrôleurs
- Disponible sur un grand nombre de processeurs
- Utilisable gratuitement pour l'enseignement

Windows

- Plusieurs versions compactes développées par Microsoft
- Windows 7E/C très utilisé dans des équipement (ex : media center)
- Windows Phone pour la téléphone mobile

LynxOS

- Développé par la société Lynx Software Technologies
- Système temps réel conforme à la norme Posix
- Utilisé dans l'avionique, l'aérospatiale, la supervision industrielle et la télécommunication.

Les principaux systèmes d'exploitation dans l'embarqué – Non linux

Nucleus

- Développé par la société Mentor Graphics
- Noyau temps réel avec une couche TCP/IP, une interface graphique, un navigateur Web et un serveur HTTP
- Livré avec les sources, pas de royalties pour la redistribution
- Très utilisé dans les terminaux bancaires de paiement électronique

NUCLEUS

VRTX

- Équipement du télescope spatial Hubble
- Gestion des processus contraints

Les principaux systèmes d'exploitation dans l'embarqué – Non linux

eCos

- Initialement développé par la société Cygnus, acquise par la Red Hat Software
- Système temps réel adapté aux solutions à très faible empreinte mémoire
- Environnement de développement basé sur Linux et chaîne de compilation GNU conforme à Posix
- Licence proche de la GPL
- Disponible par un grand nombre de processeurs
- Versions professionnelles avec support fournies par la société eCosCentric
- Utilisé dans l'industrie automobile, dans certaines imprimantes laser ou des produits multimédia

Les principaux systèmes d'exploitation dans l'embarqué – Non linux

FreeRTOS

- Développé par Amazon, Richard Barry et FreeRTOS Team
- Système temps réel adapté aux solutions à très faible empreinte mémoire
- Ordonnancement préemptif pour microcontrôleur
- Code du noyau open source, gratuit sous licence MIT (depuis 2017)
- Utilisé dans les systèmes embarqués à fortes contraintes de mémoire pour le code : réseau de capteurs sismique, robots industriels, pile réseau dans les périphériques mobiles (2012)

Les principaux systèmes d'exploitation dans l'embarqué – linux

Wind River Linux

- Édité par le leader mondiale des solutions embarquées
- Leader des solutions Linux embarqué commerciales (30 %)

WIND RIVER

MontaVista Linux

- Développé par la société MontaVista
- À l'origine des modifications du noyau Linux pour améliorer sa préemption
- Liste très fournie des processeurs supportés

MontaVista All Ways Open

BlueCat Linux

- Édité par LynxWork
- Compatibilité des exécutables sous BlueCat avec le système temps réel dur propriétaire LynxOS

Les principaux systèmes d'exploitation dans l'embarqué – linux

ELDK

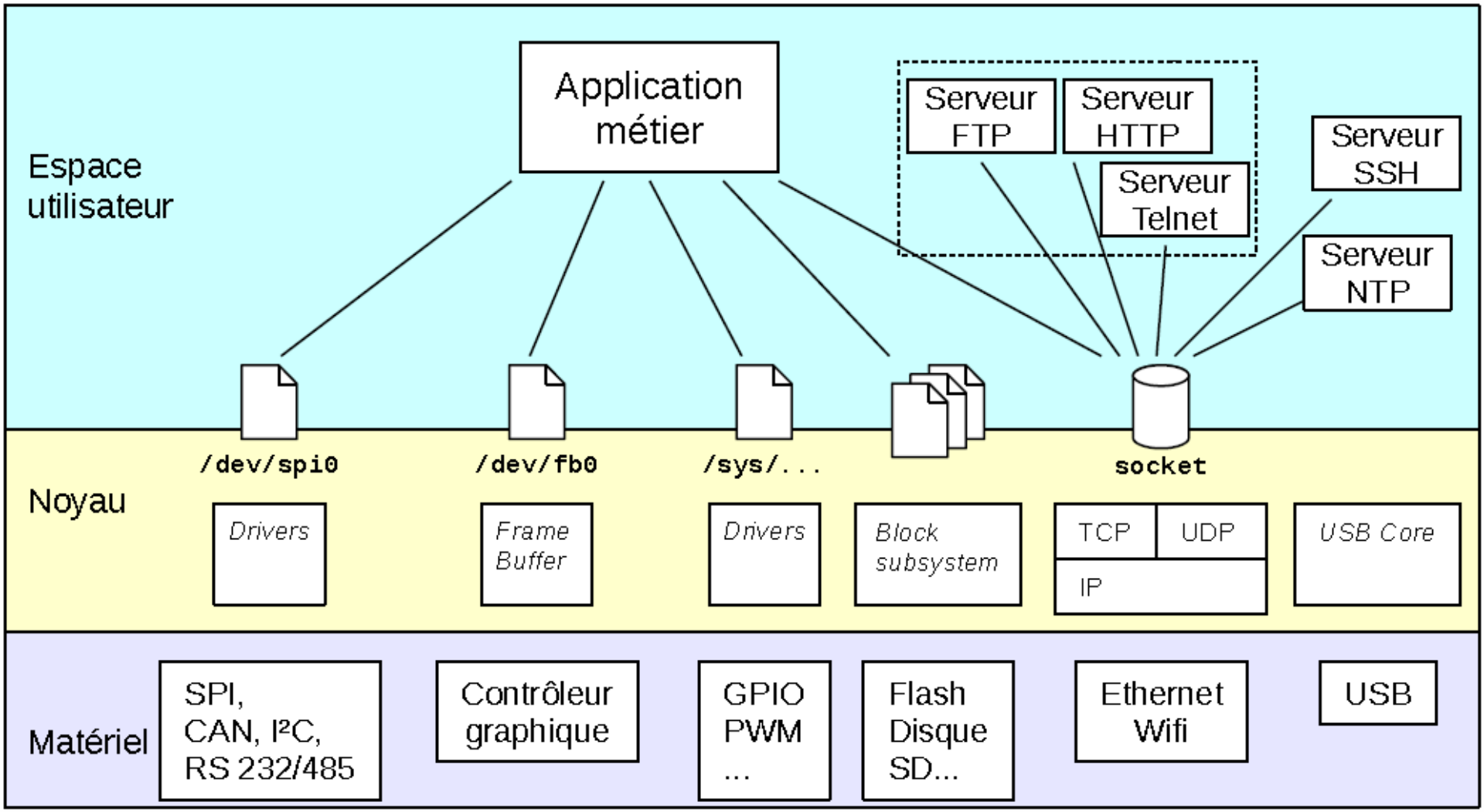
- Maintenu par la société DENX Software
- Développement du logiciel en développement croisé depuis un PC
- Linux x86 vers de nombreuses architectures
- Disponibilité d'une version complète et gratuite sous licence GPL
- Support officiel payant

µCLinux

- Version du noyau pour microcontrôleurs et processeurs sans MMU
- Disponible pour un grand nombre d'architectures de processeurs
- Utilisation dans de nombreux produits : routeurs, caméras de sécurité, lecteurs DVD ou MP3, téléphones IP, lecteurs de cartes...

Android

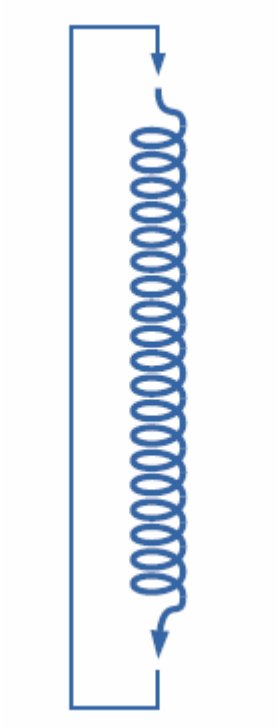
Abstraction des périphériques



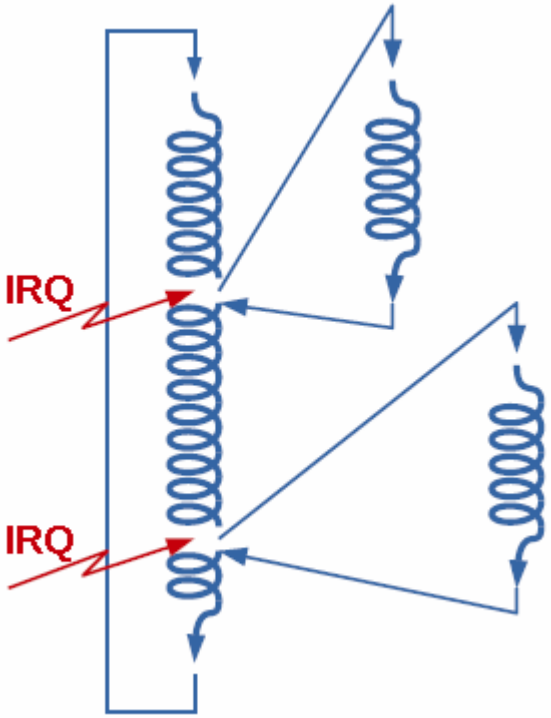
Masquer la complexité matérielle / simplifier les accès au matériel

Exécution de tâches

Système monotâche



Superloop

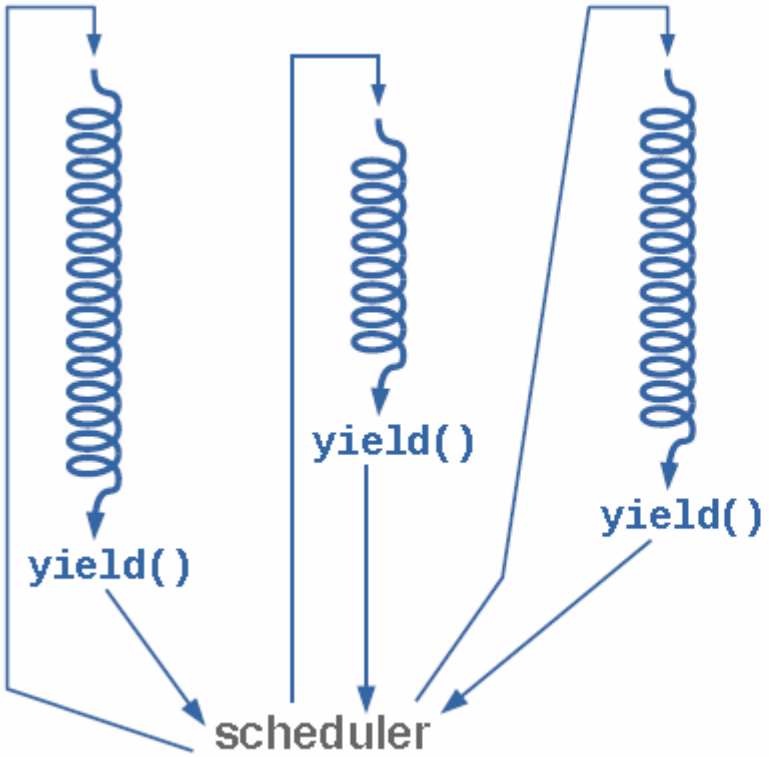


Superloop avec interruptions

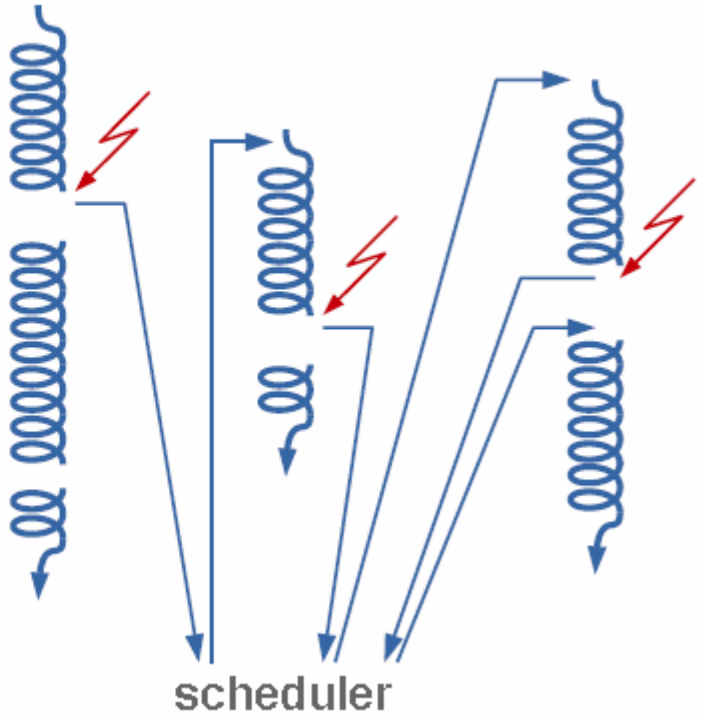
Fonctionnement typique d'un microcontrôleur / API

Exécution de tâches

Système multitâches



Coopératif



Préemptif

L'ordonnanceur (scheduler) est une fonctionnalité essentielle des systèmes OS pour exécuter des tâches sur un même processeur.

Exécution de tâches
L'ordonnancement

Tâche de sélection d'un processus en attente dans la liste des processus prêts et d'allocation de la CPU pour ce processus

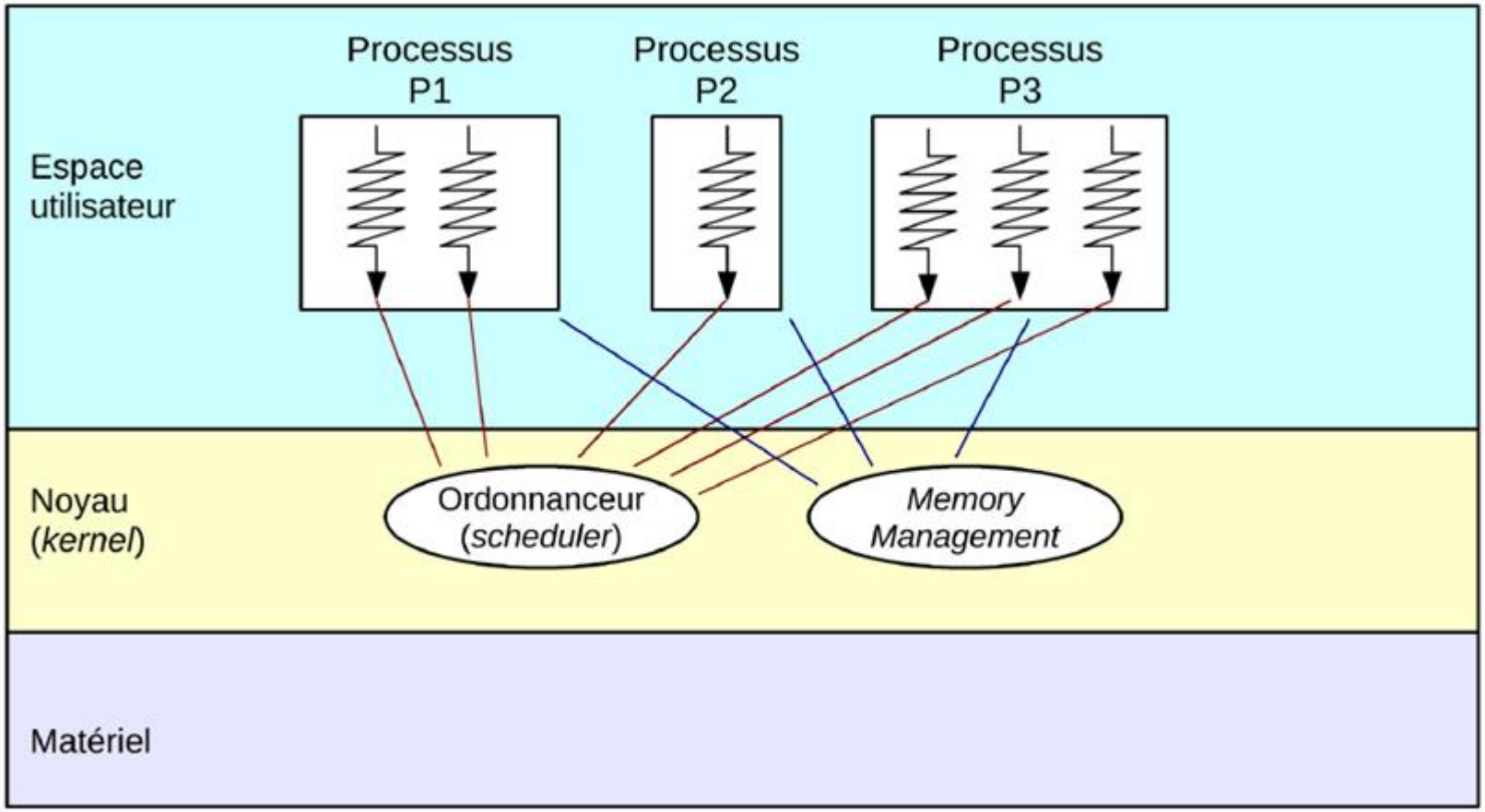
Il existe plusieurs modes d'ordonnancement :

- **Temps partagé (time sharing system)** : comportement par défaut sur les O.S. comme Linux
- **Temps réel (realtime scheduling)** : suivant des algorithmes comme Round Robin ou Fifo basés sur des priorités entre tâches ou Earliest Deadline First utilisant des temps d'expiration des tâches.

Exécution de tâches

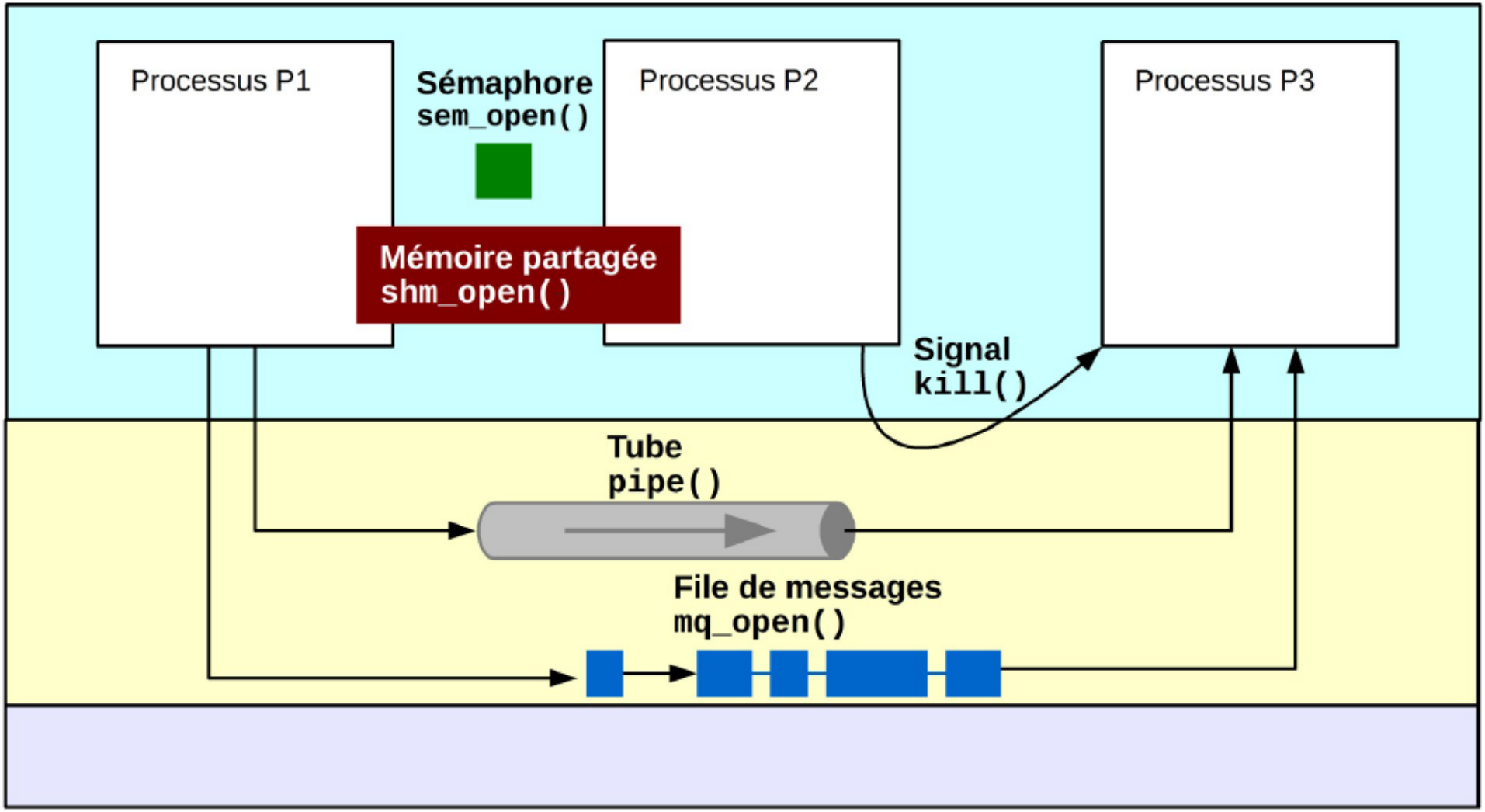
Processus et threads

Les processus sont des espaces de mémoire disjoints, au sein desquels s'exécutent un ou plusieurs threads

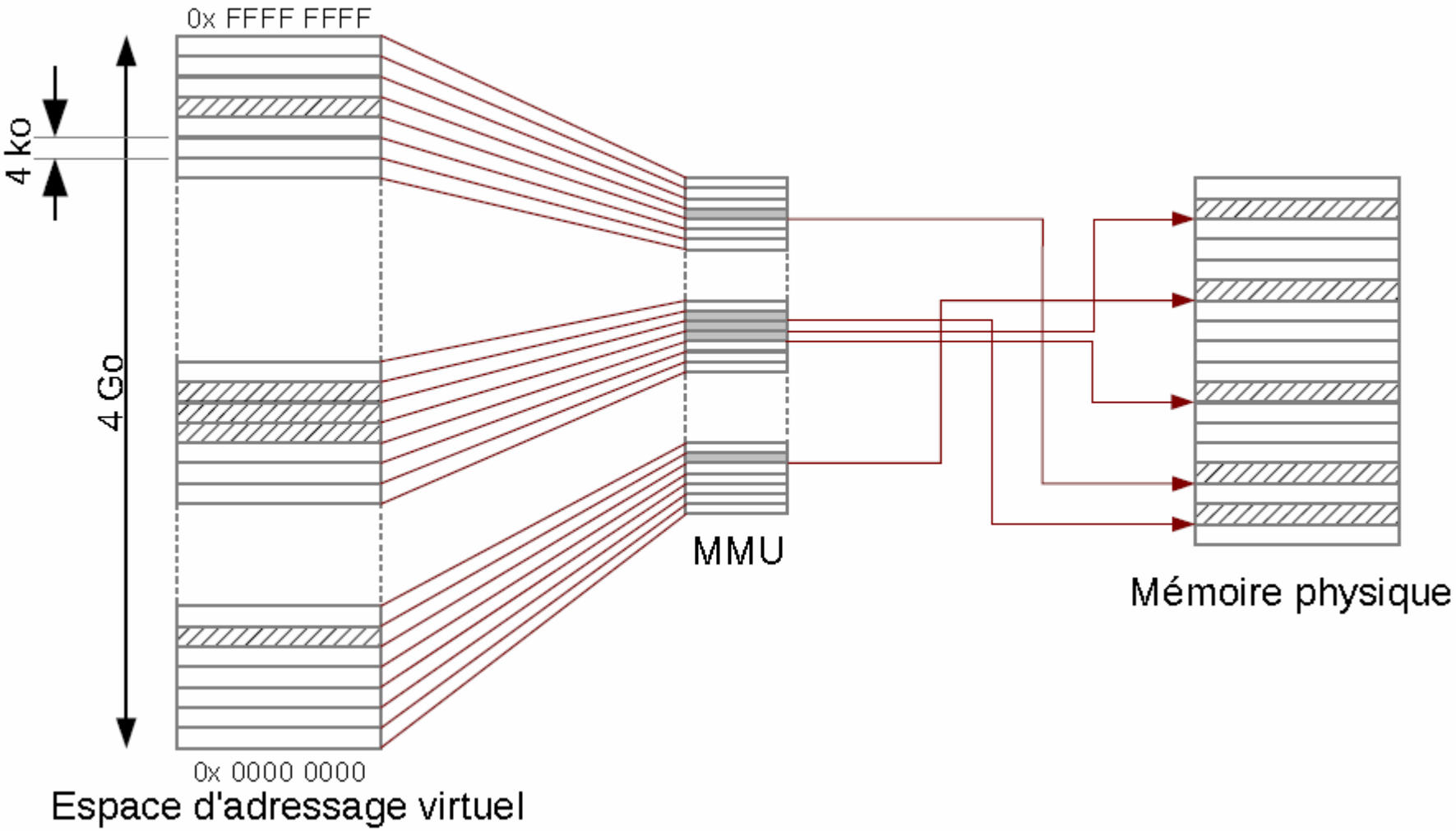


Exécution de tâches

Communication inter processus (IPC)



Mémoire virtuelle et MMU



Mémoire virtuelle et MMU

Adresses virtuelles / Adresses physiques

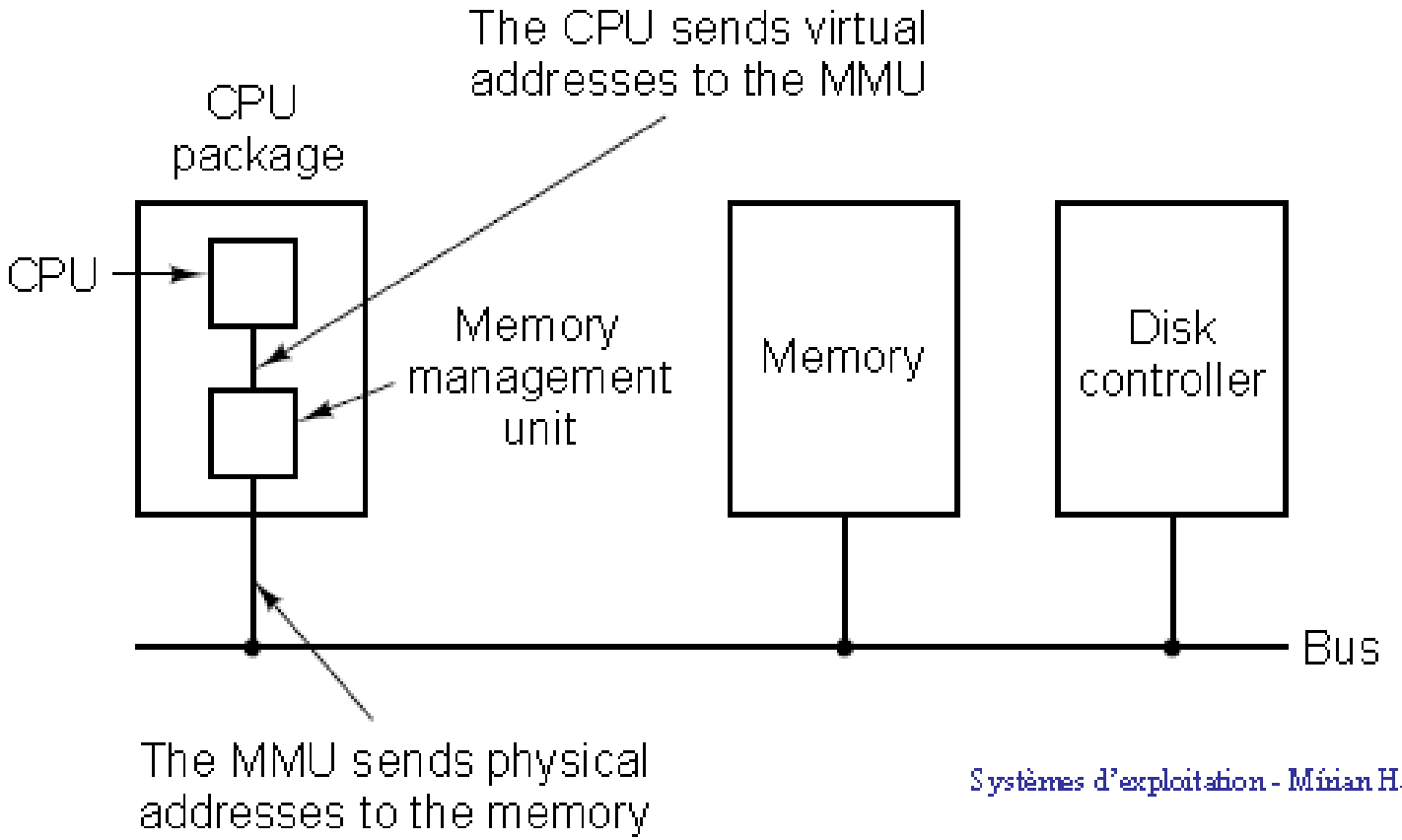
Adresse logique = adresse virtuelle :
Adresse générée par la CPU et vue par le programme utilisateur

Adresse physique :
Adresse vue par l'unité de mémoire, c-à-d chargée dans le registre d'adresse mémoire de la mémoire physique

Unité de gestion mémoire (MMU) :
Dispositif matériel intégré au microprocesseur associant les adresses logiques et physiques

Mémoire virtuelle et MMU

Principe



Systèmes d'exploitation - Mírian Halfeld-Ferrari -

Mémoire virtuelle et MMU

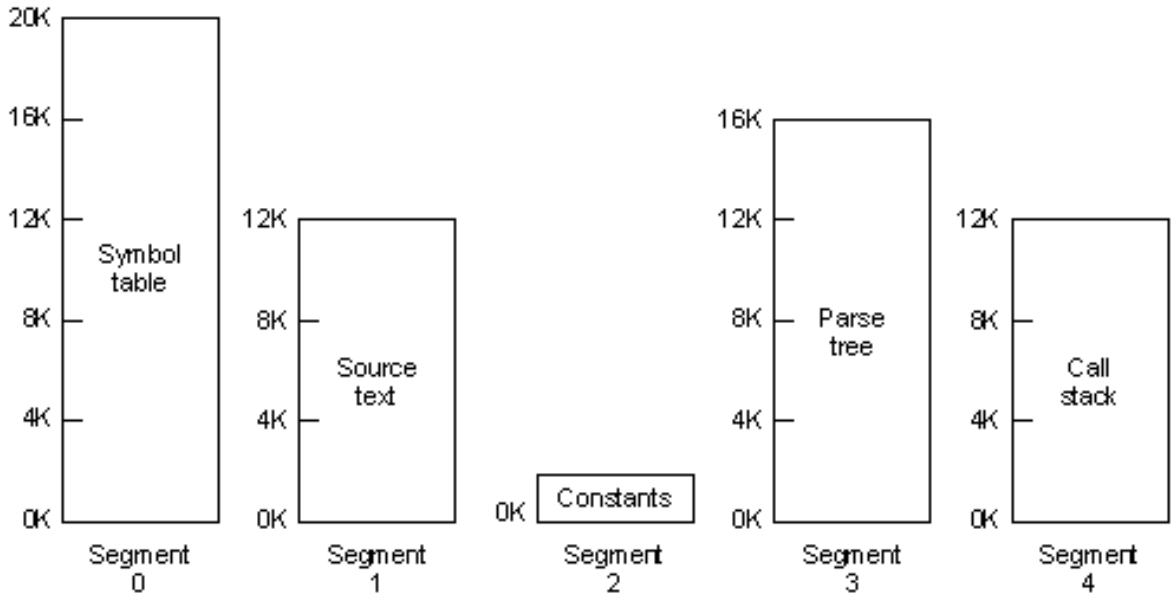
Fonctions assurées

- **Translation d'adresses** :
  - **Segmentation** : subdivision des espaces d'adressage d'après leur fonctionnalités.
  - **Pagination** : subdivision des espaces d'adressage des tâches en petites tranches de taille fixe
- **Protection** : Chaque programme reste confiné dans son espace mémoire
- **Mémoire virtuelle** : création d'un espace d'adressage important incluant la mémoire physique et une partie de la mémoire secondaire (>mémoire physique)
- **Swapping** : des portions de la mémoire sont rapatriées ou envoyées vers l'espace de stockage secondaire.

Mémoire virtuelle et MMU

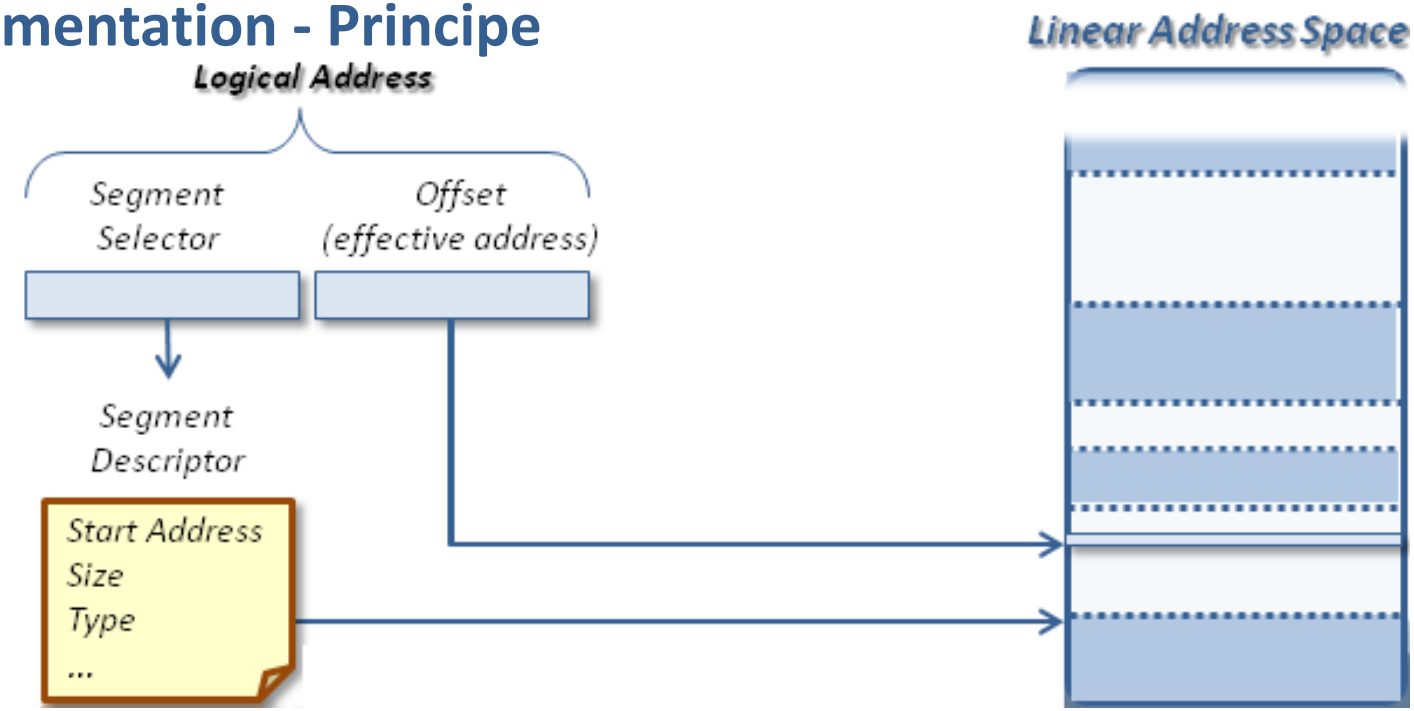
Segmentation

Un segment est une unité logique de mémoire appartenant à l'espace d'adressage d'un processus.
Les segments sont de tailles variables et contiennent des données du même type.



Mémoire virtuelle et MMU

Segmentation - Principe



L'adresse logique se calcule en additionnant :

- le contenu du registre correspondant **au numéro de segment**
- le **déplacement** contenu en partie basse de l'adresse

une violation de protection mémoire est détectée si le déplacement atteint ou excède la **limite** du segment

Mémoire virtuelle et MMU

Segmentation - Exercice

On considère la table des segments suivante pour un processus P1 :

Index	Base	Limite
0	540	234
1	1254	128
2	54	328
3	2048	1024
4	976	200

Calculez les adresses physiques correspondant aux adresses logiques suivantes. Signalez éventuellement les erreurs de violation.

(0:128) :

(1:100) :

(2:465) :

(3:888) :

(4:100) :

(4:344) :

Mémoire virtuelle et MMU

Segmentation - Protection et partage

Protection :

- Association de la protection avec le segment
- bits de protection associés à chaque entrée de la table de segments pour empêcher des accès illégaux à la mémoire

Partage :

- Les segments sont partagés quand les entrées dans les tables de segments de 2 processus différents pointent vers les mêmes emplacement physiques
- Chaque processus possède une table de segments

Mémoire virtuelle et MMU

Segmentation - Protection et partage

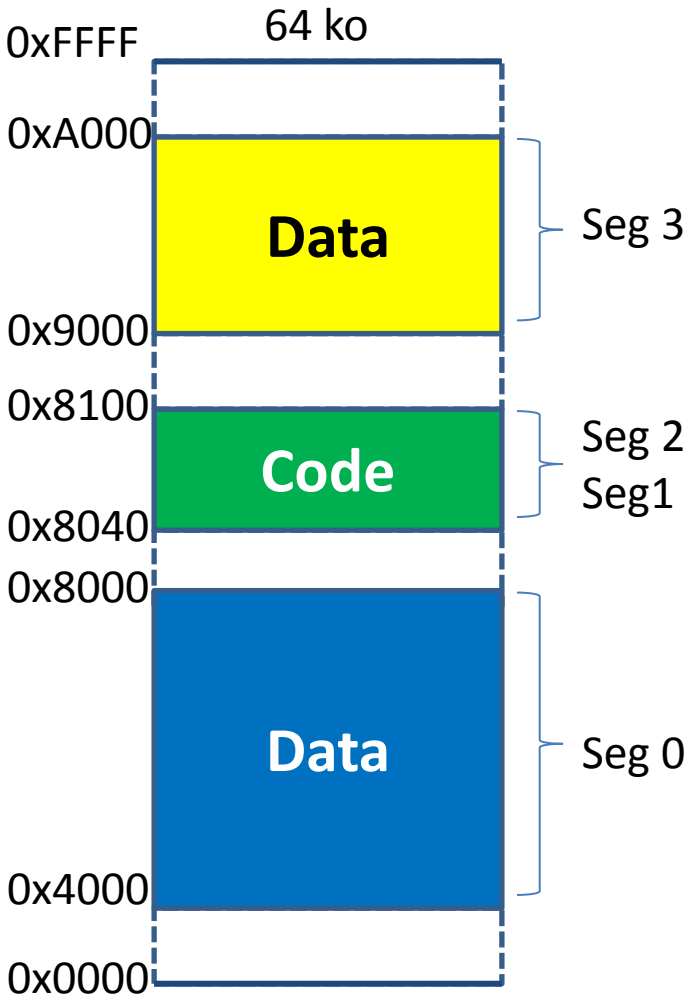


Table des segments du processus 1

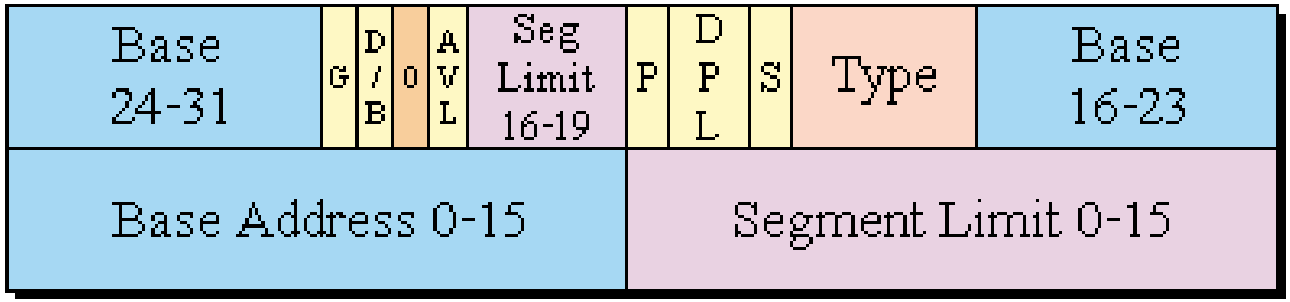
Index	Base	Limite	Type	Priv.
0	0x4000	0x3FFF	2	3
1	0x8040	0x00BF	10	3

Table des segments du processus 2

Index	Base	Limite	Type	Priv
2	0x8040	0x00BF	10	3
3	0x9000	0x0FFF	2	3

Mémoire virtuelle et MMU

Segmentation – Descripteur de segment (Intel 80x86)



AVL - Available for use by the operating system
BASE - Segment Base Address
D / B - Default Segment Size (16 / 32 bits)
DPL - Descriptor Privilege Level
G - Granularity
LIMIT - Segment Limit
P - Present Bit
S - Descriptor Type (System / Application)
TYPE - Segment Type

Type bits for Data segments

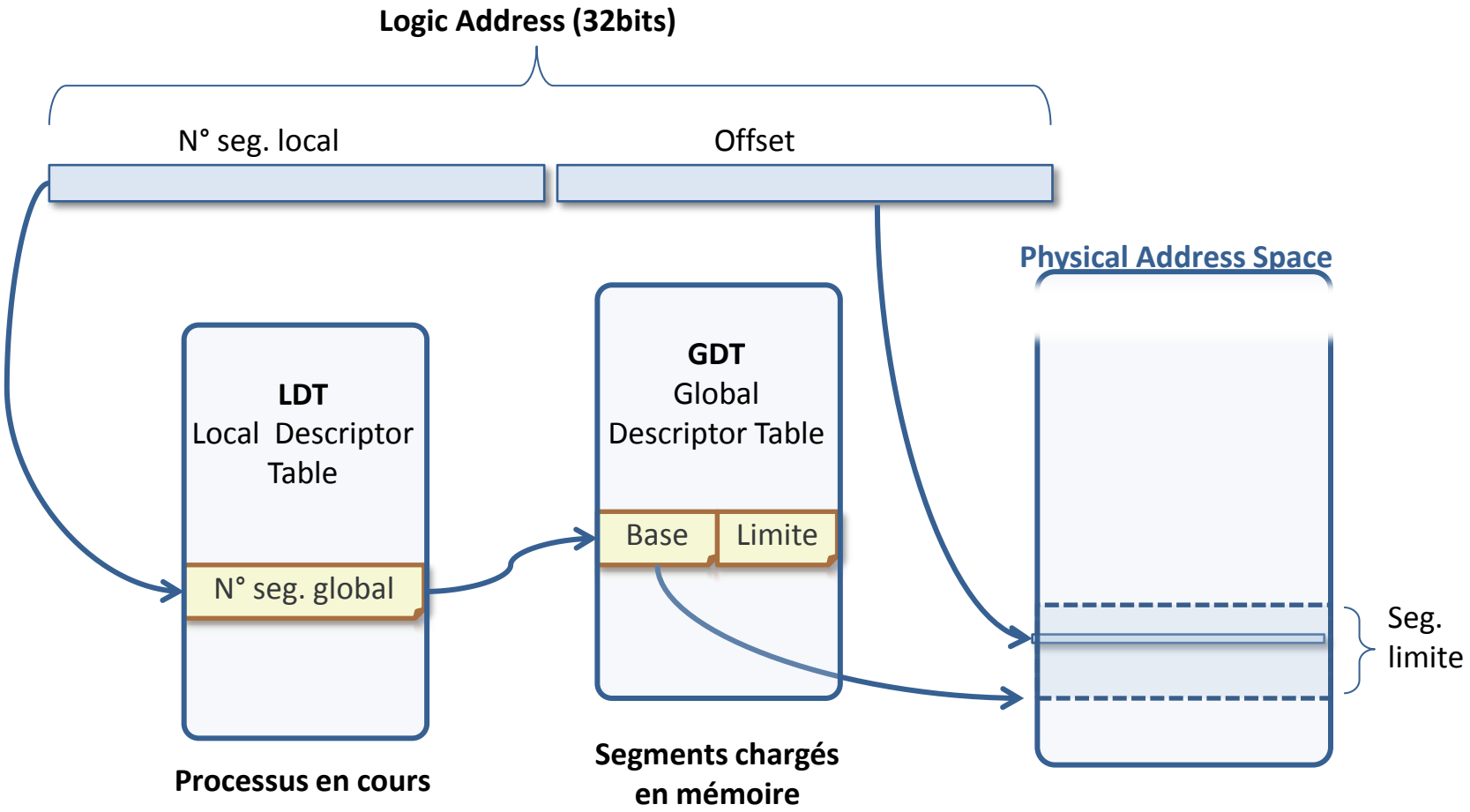
bit 3	Data/Code	0 (data)
bit 2	Expand-down	0 (normal) 1 (expand-down)
bit 1	Writable	0 (read-only) 1 (read-write)
bit 0	Accessed	0 (hasn't been accessed) 1 (has been accessed)

Type bits for Code segments

bit 3	Data/Code	1 (code)
bit 2	Conforming	0 (non-conforming) 1 (conforming)
bit 1	Readable	0 (execute-only) 1 (executable and readable)
bit 0	Accessed	0 (hasn't been accessed) 1 (has been accessed)

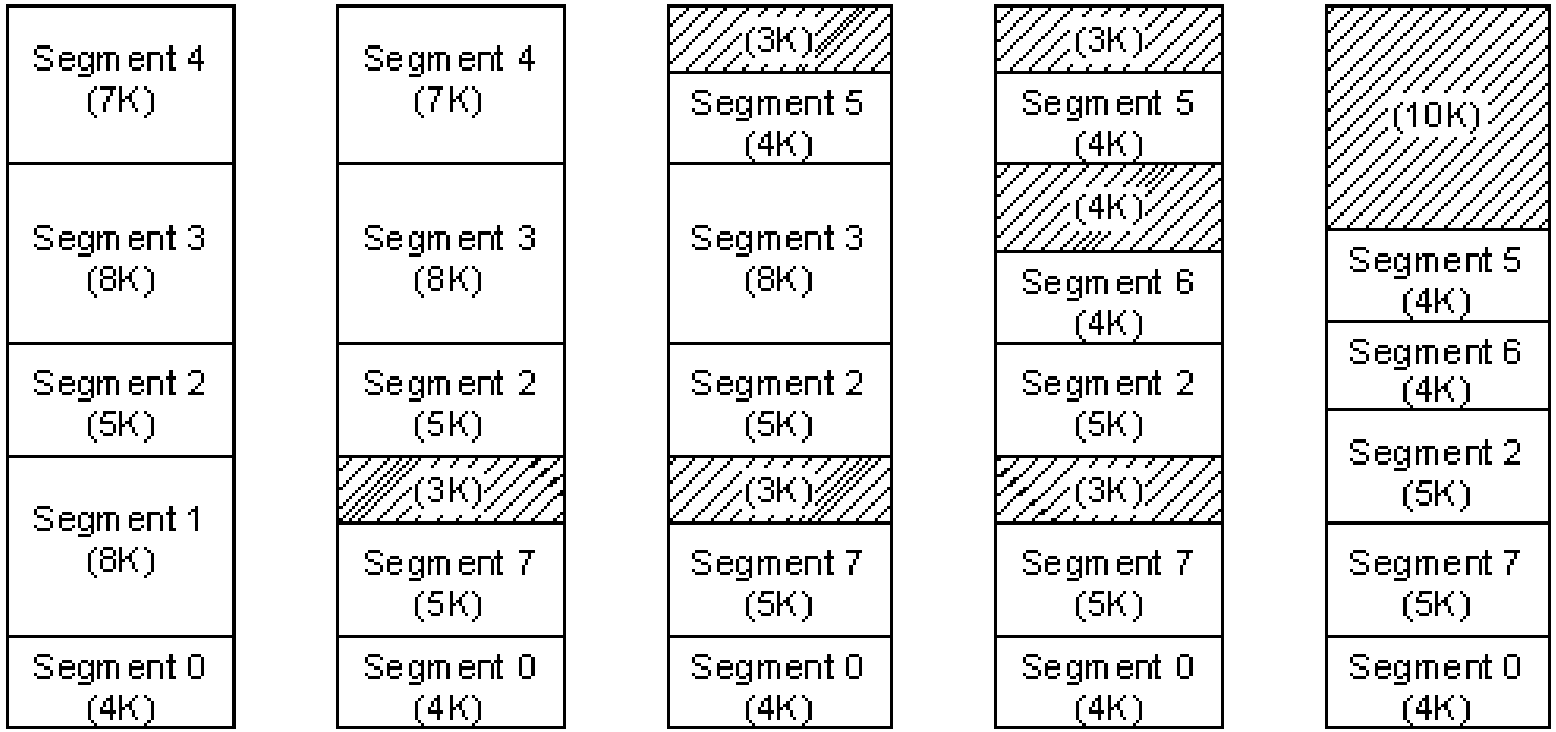
Mémoire virtuelle et MMU

Segmentation - Tables locale/globale des segments



Mémoire virtuelle et MMU

Segmentation - Fragmentation

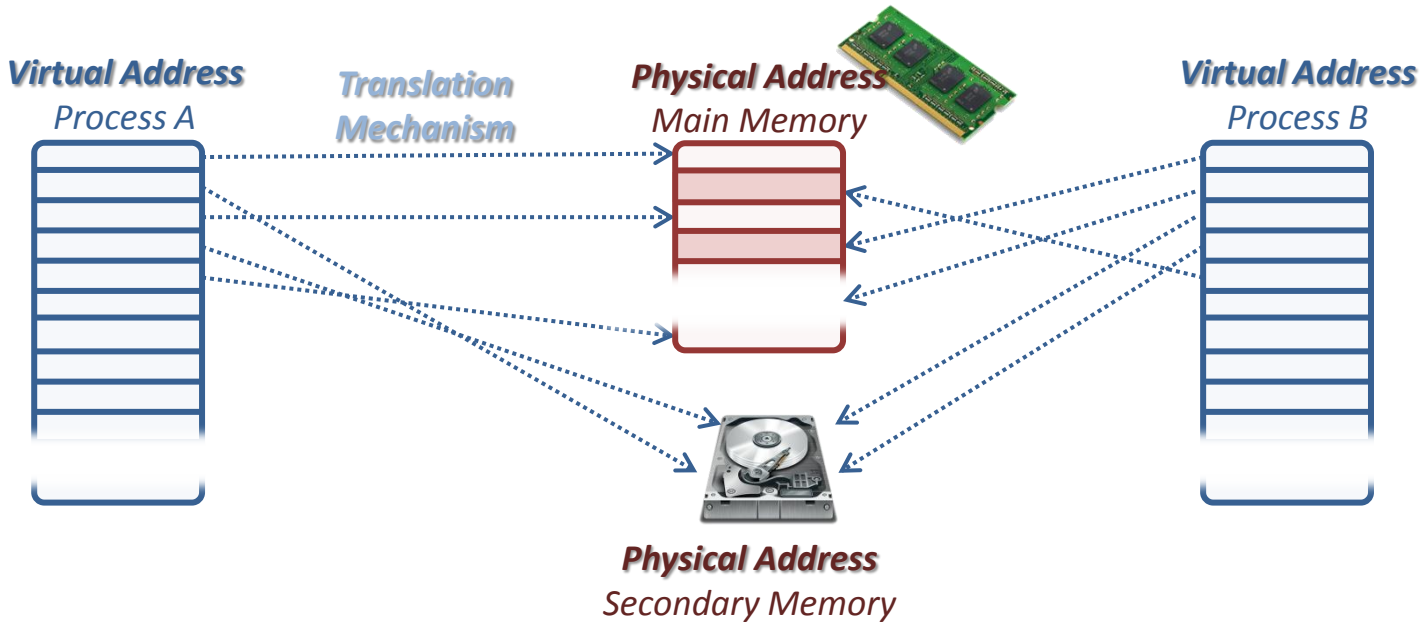


L'inconvénient majeur de la segmentation est la fragmentation externe de la mémoire physique.

Mémoire virtuelle et MMU

Pagination

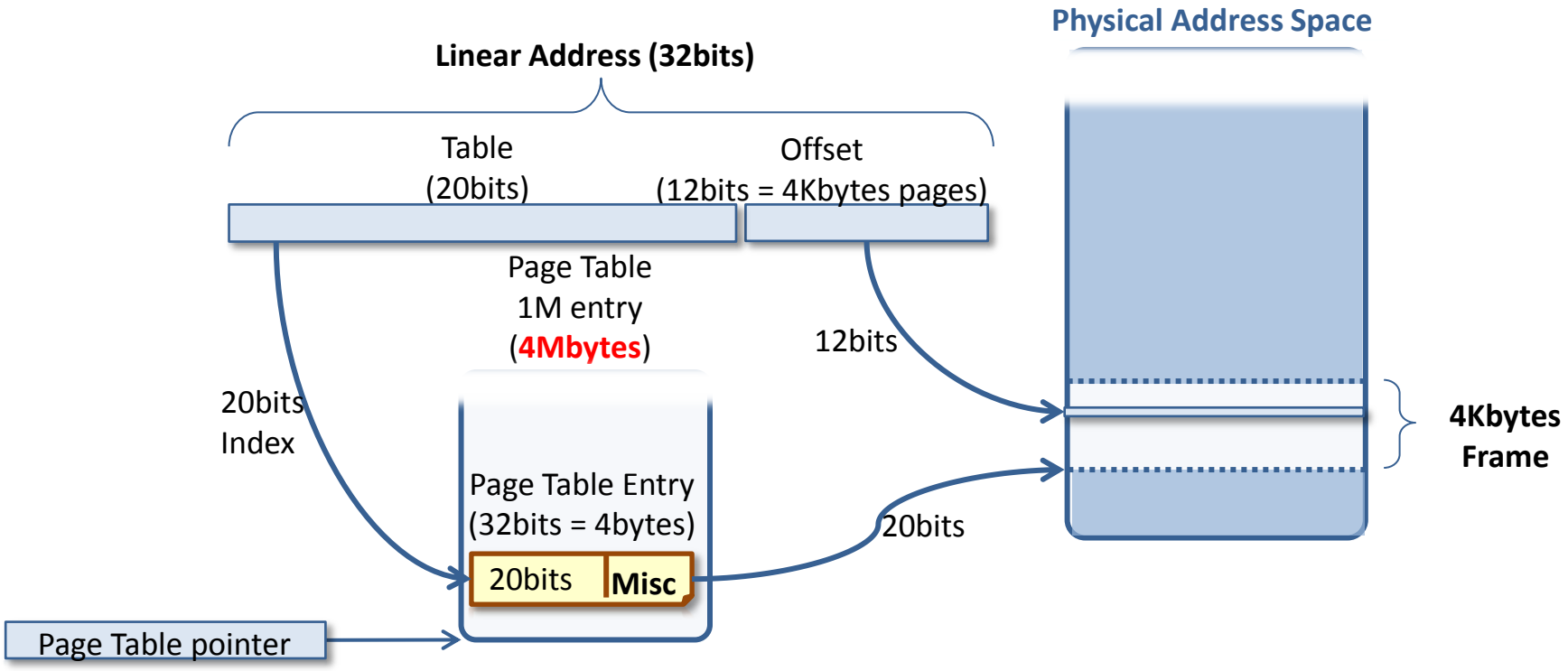
Une mémoire paginée est découpée en pages fixant ainsi une granularité de la mémoire. La mémoire principale est découpée en frames (cadres) de même taille, chaque cadre contenant une page. Il peut y avoir plus de pages que de cadres



Mémoire virtuelle et MMU

Pagination

Mécanisme de translation d'adresses (mémoire linéaire vers mémoire physique) de l'unité de pagination est extrêmement performant et consiste à une simple consultation d'une table des pages



Mémoire virtuelle et MMU

Pagination - Exercice

Dans un système paginé :

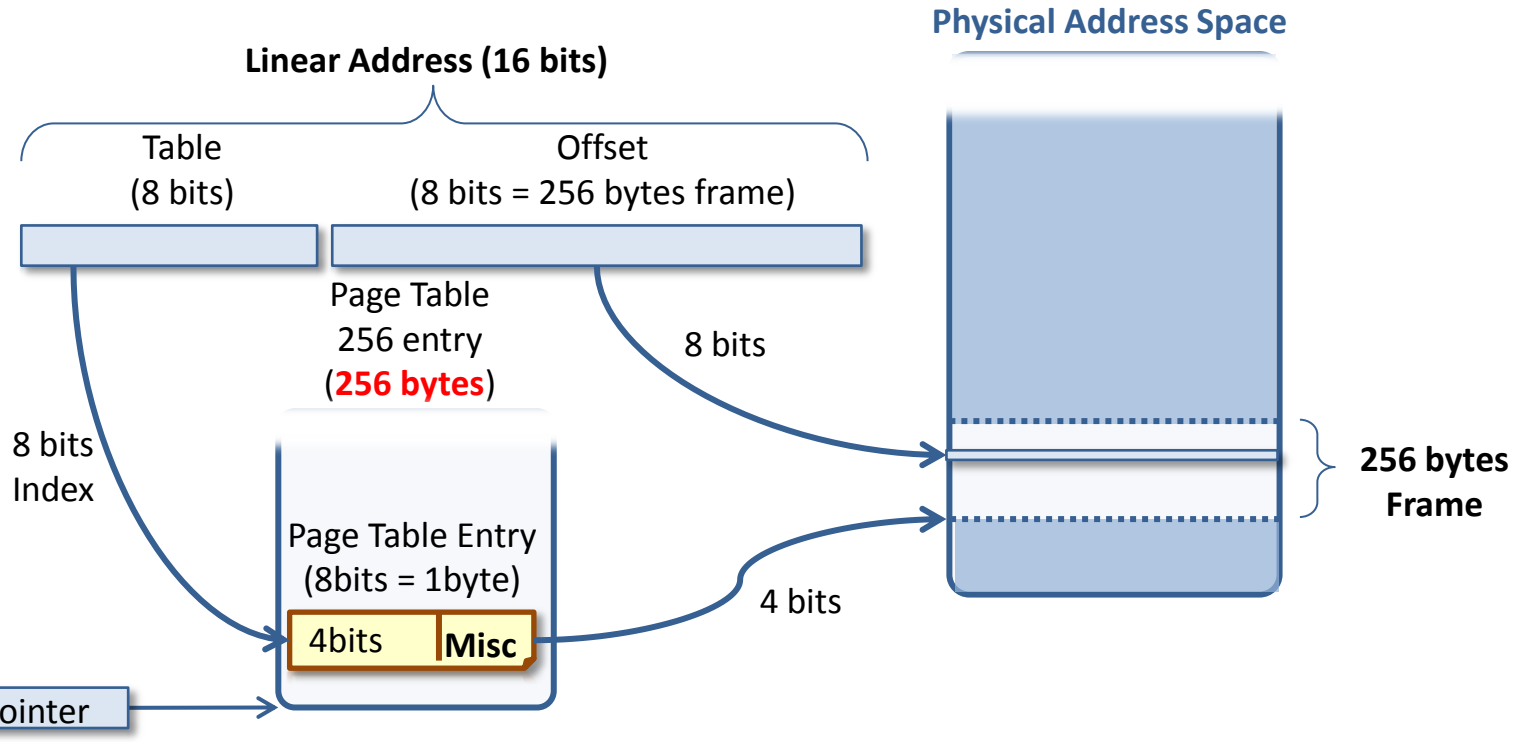
- Les pages ont une taille de 256 octets
- On autorise chaque processus à utiliser au plus 4 cadres
- Les adresses physiques sont codées sur 12 bits
- Les adresse virtuelles sont composées de 8 bits pour la page et 8 pour l'offset

On considère la table des pages suivante d'un processus P1 :

Page	0	1	2	3	4	5	6	7
Cadre	0011	0001	0000	0010	0100	0111	0101	0110
Présence	1	0	1	0	0	0	1	0

1. **Quelle est la taille de l'espace d'adressage du processus P1 ?**
2. **De combien de mémoire vive dispose ce système ?**
3. **Calculez les adresses réelles correspondant aux adresses virtuelles suivantes : 240, 546, 1761, 2539**
4. **Que se passe-t-il si P1 génère l'adresse virtuelle 783 ?**

Mémoire virtuelle et MMU
Pagination - Exercice



@V : 240 => 0000 0000 1111 0000
n° Page = 0 => n° Cadre = 0011
Offset = 240 => 1111 0000
@R : 0011 1111 0000 => 1008

@V : 1761 => 6 x 256 + 225
n° Page = 6 => n° Cadre = 0101
Offset = 225
@R : 5 x 256 + 225 = 1505

@V : 546 => 0x0222
n° Page = 0x2 => n° Cadre = 0000
Offset = 0x22
@R : 0x0022 => 34

@V : 2539 => Page 9 : Hors de l'espace virtuel d'adressage du processus

@V : 783 => Page 3 / Présence = 0 : Défaut de page

Mémoire virtuelle et MMU
Pagination

Problèmes majeurs de la construction des ordinateurs:

- La table des pages est extrêmement grande.
  - Ordinateurs modernes : adresses virtuelles d'au moins 32 bits.

  => Table de pages avec plus de un million d'entrées!
- Chaque processus a besoin de sa propre table de pages.
  - La correspondance doit être rapide.
  - Pour chaque instruction il est nécessaire de faire référence à la table des pages 1 fois, 2 fois et parfois plus.

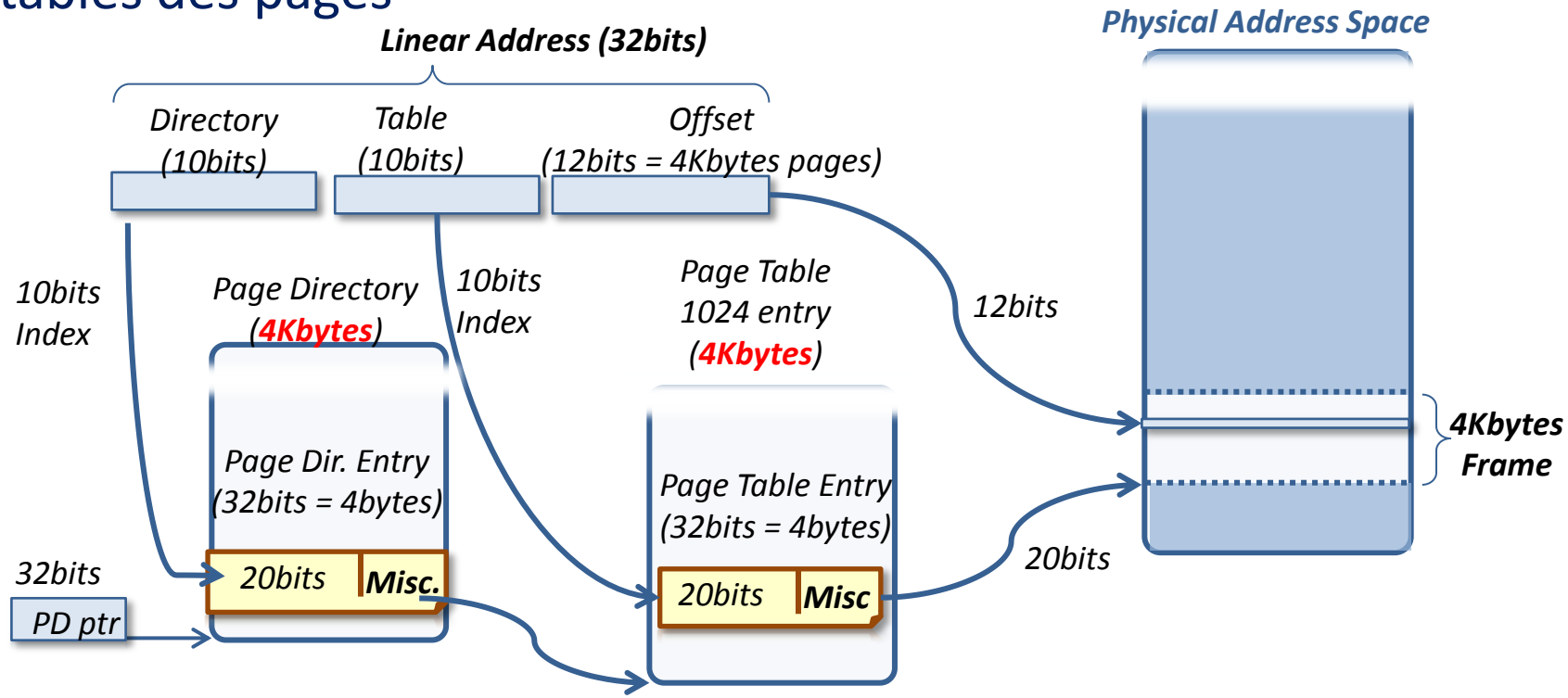
Mémoire virtuelle et MMU

Pagination

Problème du mécanisme de pagination :

- taille de la table des pages (4Mo).
- table est présente en mémoire principale.

Solution : utiliser une seconde table sauvant des pointeurs vers la tables des pages



Mémoire virtuelle et MMU

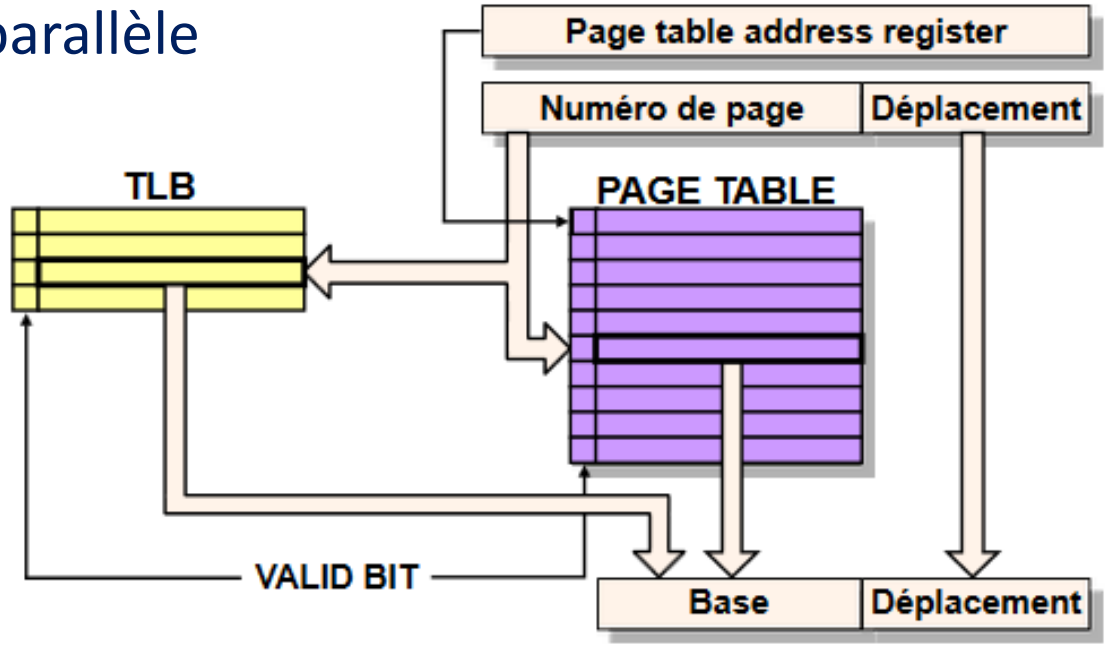
Pagination

Problème du mécanisme de pagination :

- Accès répétés aux mêmes données en mémoire

Solution :Utiliser les registres associatifs (ou Translation Lookaside Buffers: TLB)

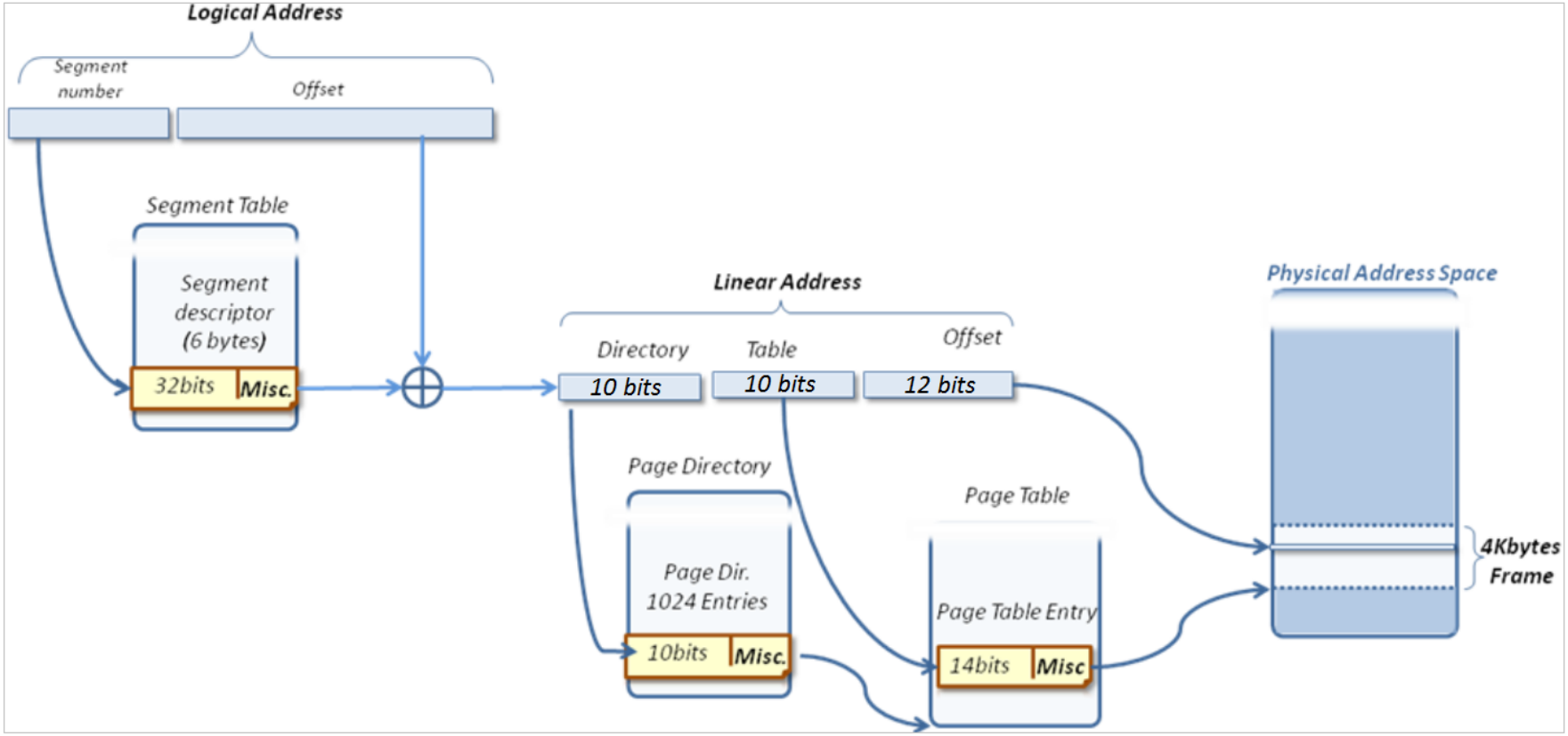
- Petite mémoire cache matérielle spéciale à la consultation rapide.
- Contiennent seulement quelques entrées de la table
- Recherche en parallèle



Mémoire virtuelle et MMU : Segmentation et Pagination

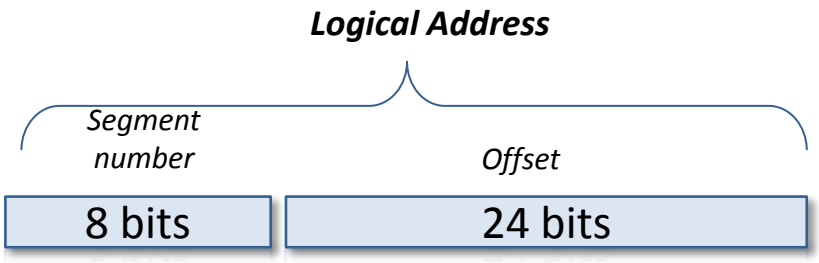
Soit un système de gestion de mémoire gérée de manière segmentée et paginée avec double niveau de pagination.

- La taille de la mémoire physique est de 64 Mo (1mot = 1 octet).
- Un processus peut avoir au plus 256 segments.
- Chaque segment peut adresser au plus 16 Mo.
- La taille d'une page est fixée à 4 ko.



Mémoire virtuelle et MMU : Segmentation et Pagination

- Quel est le format des adresses logiques ? Expliquez.
  Une adresse logique est composée d'un n° de segment et d'un décalage.
  - Le numéro de segment doit pouvoir représenter 256 segments => 8 bits
  - Le décalage doit pouvoir adresser un espace de 16 Mo => 24 bits



- Quel est le format d'une adresse physique ? Expliquez.
  L'espace d'adressage physique est de 64 Mo => 26 bits.

- Quelle est la taille de l'espace d'adressage virtuelle
  1 adresse virtuelle = 32 bits donc espace d'adressage de 4 Go ou 256 segments de 16 Mo au max = 4 Go.

Mémoire virtuelle et MMU : Segmentation et Pagination

Soit un processus muni de la table des segments suivante :

Segment	Base	Limite
00	00 BE 0A 00	10 00
01	00 BE 23 D1	02 FF
02	00 BE 00 DA	03 61
03	00 BE 1A 26	05 07
04	00 BE 0F F0	10 00

Du répertoire de pages suivant :

Répertoire	Table	Active
0	0	1
1	3	0
2	1	1
3	2	0

Et de deux tables de pages :

N°	@Page	@Cadre	Active	Libre
0	2A0	23 40	1	0
1	2A1	05 BB	1	0
2	2A2	00 00	0	0
3	2A3	14 E0	1	0
4				1
				1
1023				1

Table 0

N°	@Page	@Cadre	Active	Libre
0	3E0	00 00	0	0
1	3E1	27 FD	1	0
2	3E2	00 00	0	0
3	3E3	3A F6	1	0
4				1
				1
1023				1

Table 1

Mémoire virtuelle et MMU : Segmentation et Pagination

- Quelle est l'adresse linéaire correspondante à l'adresse logique **0x030000F0**

N° de segment : 0x03

=> adresse de base : **0X00BE1A26**

Décalage : **0x0000F0**

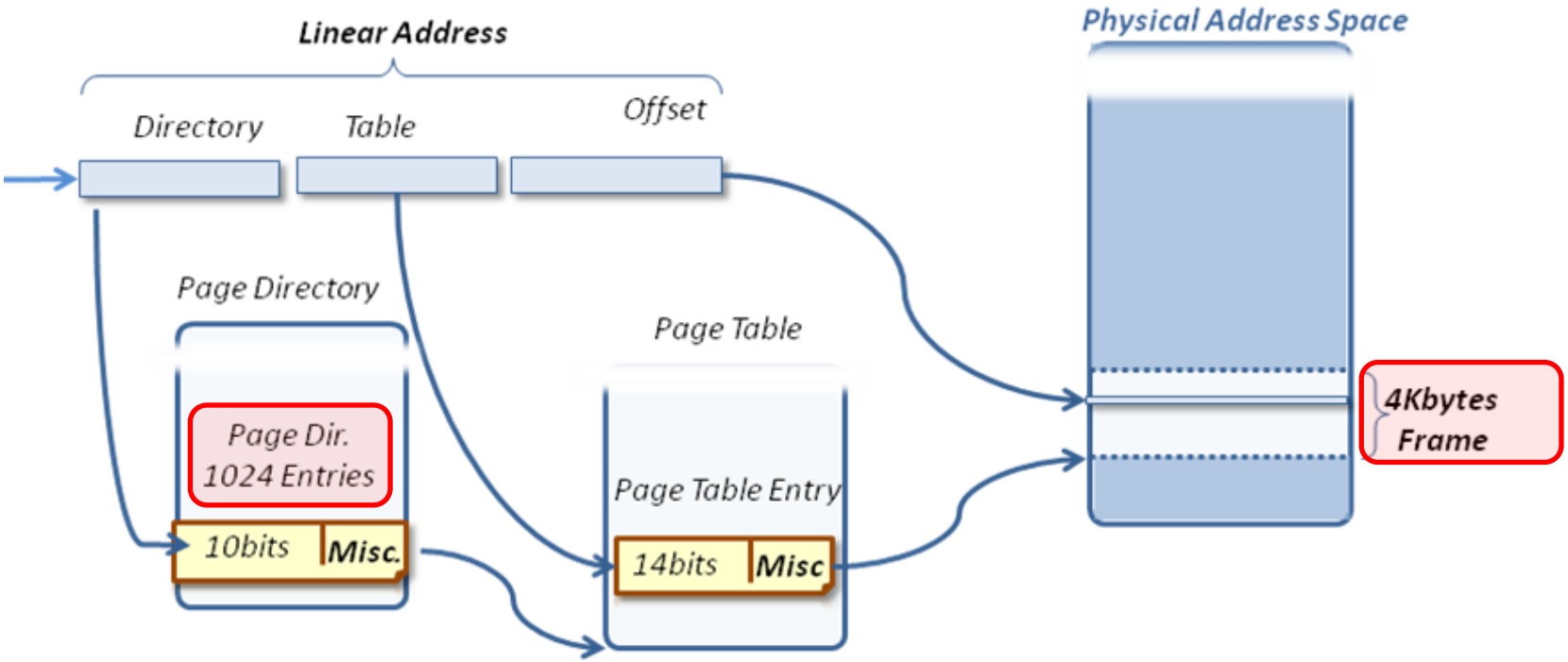
=> inférieur à la limite du segment (0x0507) **pas d'erreur de dépassement**

Adresse linéaire = **0x00BE1A26 + 0x0000F0 = 0x00BE1B16**

Segment	Base	Limite
00	00 BE 0A 00	10 00
01	00 BE 23 D1	02 FF
02	00 BE 00 DA	03 61
03	00 BE 1A 26	05 07
04	00 BE 0F F0	10 00

Mémoire virtuelle et MMU : Segmentation et Pagination

- Quel est le format d'une adresse linéaire ? Expliquez.
  - Le répertoire de page contient 1024 entrées => **10 bits pour identifier une entrée**
  - Une page contient 1024 entrées => **10 bits pour identifier une entrée**
  - La taille d'une page est de 4 ko => **12 bits de décalage**



Mémoire virtuelle et MMU : Segmentation et Pagination

- Quelle est l'adresse physique correspondante à l'adresse logique **0x030000F0**

Adresse logique **0x030000F0**<=> Adresse linéaire **0X00BE1B16**

0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0	0	0	0	1	1	0	1	1	0	0	0	1	0	1	1	0
Répertoire : 2										Page : 3E1										Décalage : B16											

Répertoire	Table	Active
0	0	1
1	3	0
2	1	1
3	2	0

N°	@Page	@Cadre	Active	Libre
0	3E0	00 00	0	0
1	3E1	27 FD	1	0
2	3E2	00 00	0	0
3	3E3	3A F6	1	0
4				1
				1
1023				1

Table 1

Adresse Physique = 26 bits : @page+décalage

=> **0x27FDB16**

Mémoire virtuelle et MMU : Segmentation et Pagination

- Quelle quantité de mémoire physique occupe le processus

  - 1 table de segment contenant 5 descripteurs de segments de 6 octets : **30 octets**

  - 1 répertoire de table contenant 4 numéros de table et leur état d'activité.
  - Numéro de table codé sur 10 bits et état d'activité sur 1 bit : **2 octets**.

  => Mémoire physique occupée par le répertoire de table et la table des segments :
  30 + 4x2 = **38 octets (négligeable)**

  - 2 tables contenant 1024 entrées chacune.
  - Numéro de cadre codé sur 14 bits + 1 bit d'activité + 1 bit de liberté : Nécessite 2 octets

  $\Rightarrow$ Quantité de mémoire physique occupée par les tables : 2x2x1024 = **4 ko**
  $\Rightarrow$ 5 pages actives et une page occupe 4 ko : 5x4 = **20 ko**
  $\Rightarrow$ Occupation totale en mémoire physique : **24 ko**

Mémoire virtuelle et MMU

Pagination

Linux depuis la version 2.6.11 : Pagination à 4 niveaux/pages de 4Ko

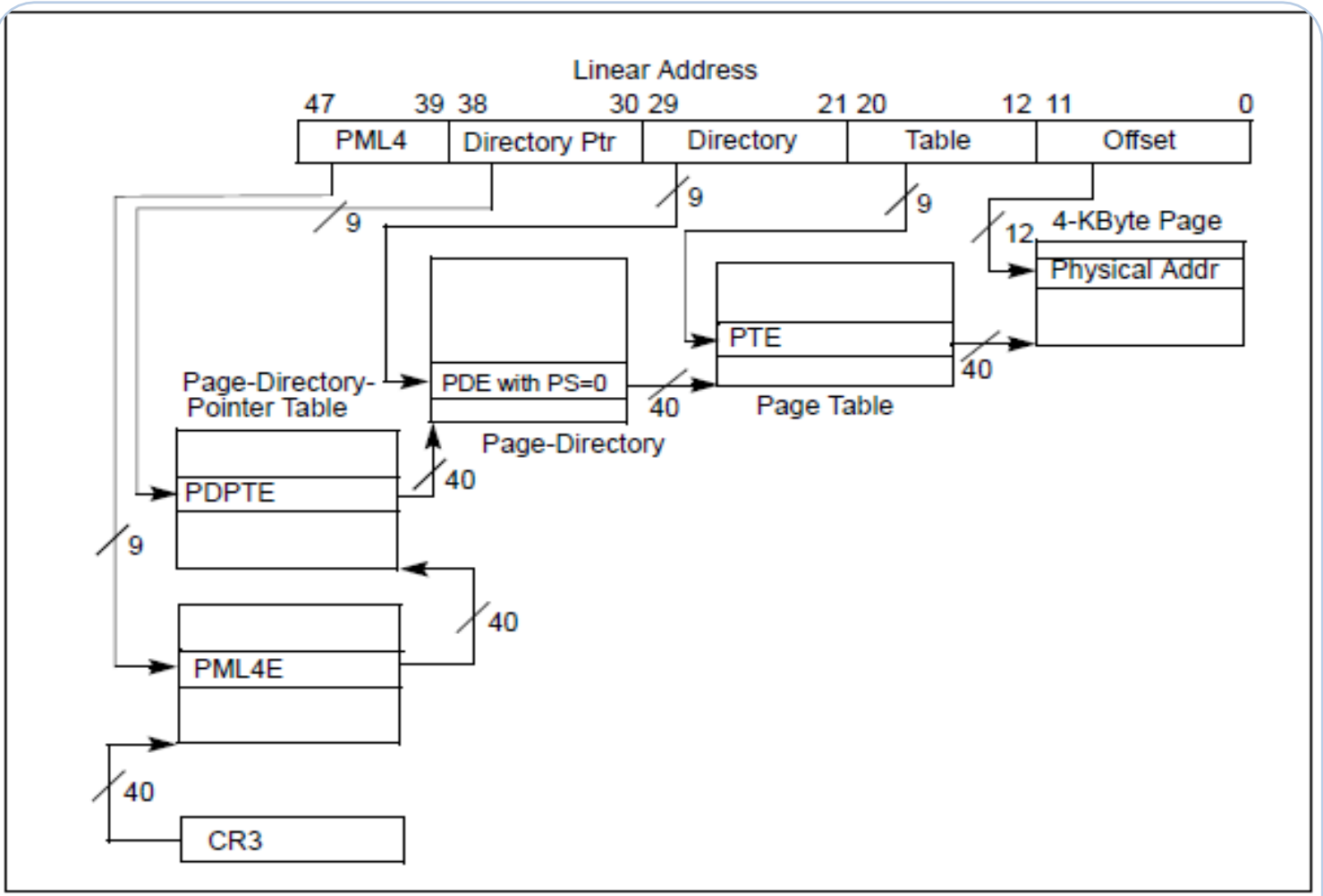
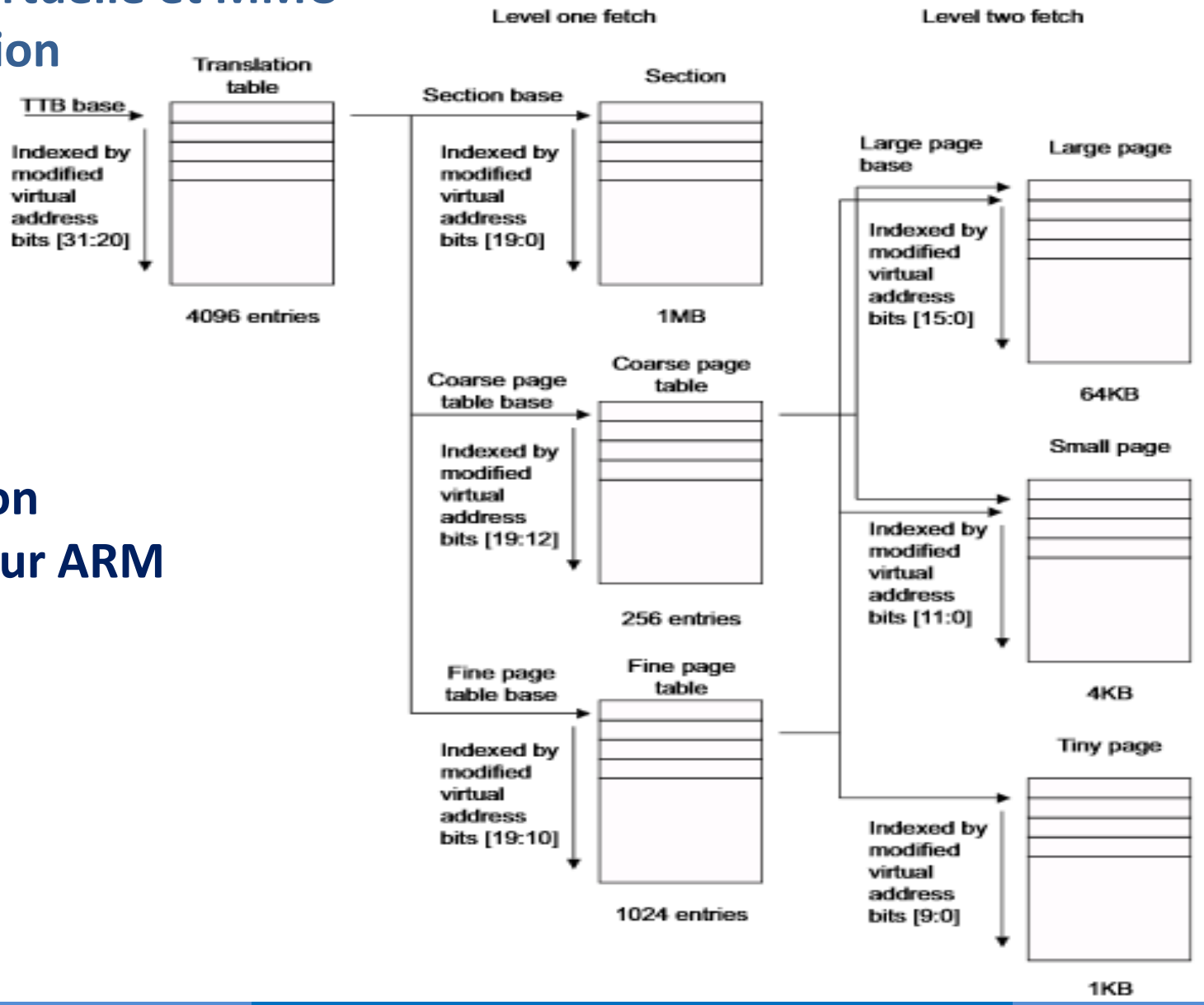


Figure 4-8. Linear-Address Translation to a 4-KByte Page using IA-32e Paging

Mémoire virtuelle et MMU

Pagination



Pagination processeur ARM

- **Abraham Silberschatz, Peter Baer Galvin, Greg Gagne : Operating System Concepts (2011)**
- **Andrew Tanenbaum : Systèmes d'exploitation (3ème édition, 2008)**
- **Gilles Blanc : Linux embarqué (2011)**
- **Pierre Ficheux : Linux embarqué : (3è édition, 2010)**
- **Cours de Stéphane Huet : Principes des OS / Linux embarqué (2014)**
- **Christophe Blaess : Ingéniérie et formations sur les systèmes libres** http://www.blaess.fr/christophe/
- **Cours de Jalil Boukhobza : Systèmes d'exploitation pour l'embarqué** http://syst.univ-brest.fr/~boukhobza/index.php/systemes-dexploitation-pour-lembarque
- **Cours de Hugo Descoubes : Architecture des ordinateurs** https://www.canal-u.tv/producteurs/centre_d_enseignement_multimedia_universitaire_c_e_m_u/ensicaen/architecture_et_technologie_des_ordinateurs
- **Free electron : Formation Buildroot** http://free-electrons.com/doc/training/buildroot/buildroot-slides.pdf
- **Tutoriel Premiers pas avec Xenomai : David CHABAL** http://dchabal.developpez.com/tutoriels/linux/xenomai/