



Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes Embarqués 1 & 2

p.04 - Architecture générale

Classes T-2/I-2 // 2018-2019

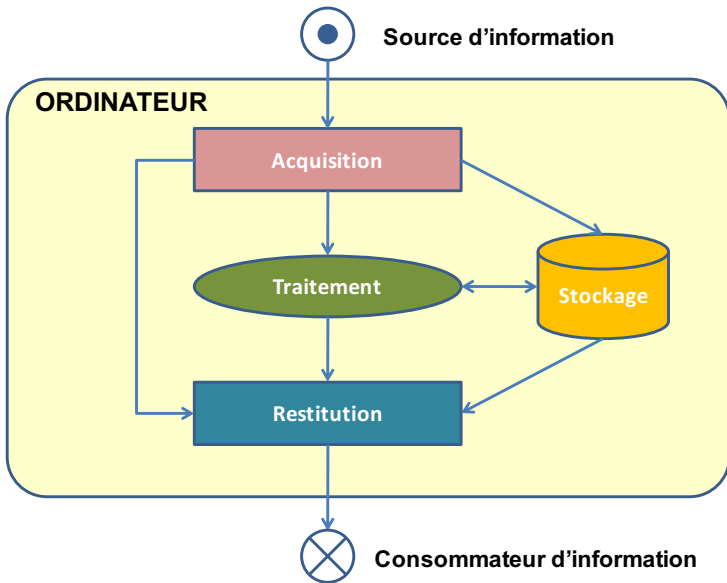
Daniel Gachet | HEIA-FR/TIC
p.04 | 09.04.2019



- Introduction
- Architectures des microprocesseurs
- Mémoire principale
- Représentation de l'information
- Bus microprocesseur

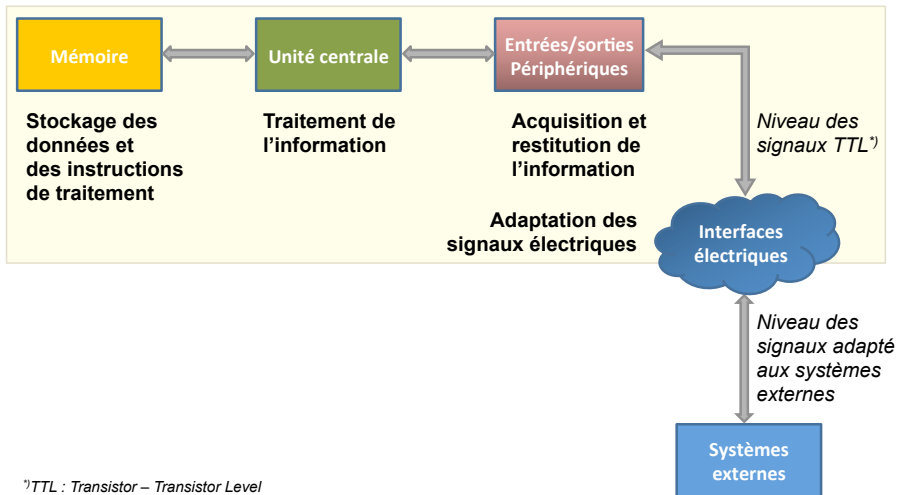


Architecture générale des ordinateurs





Architecture générale des systèmes à microprocesseurs

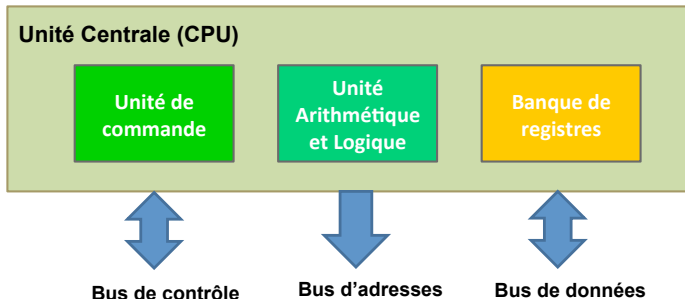


^{*)}TTL : Transistor – Transistor Level



Architecture générale de l'unité centrale

- L'unité centrale (CPU - *Central Processing Unit*) est composée
 - ▶ d'une unité de commande (exécution des instructions)
 - ▶ d'une unité arithmétique et logique (ALU - *Arithmetic and Logical Unit*)
 - ▶ d'une banque de registres
 - ▶ d'un bus pour l'échange d'information avec les périphériques (contrôle, adresses et données)





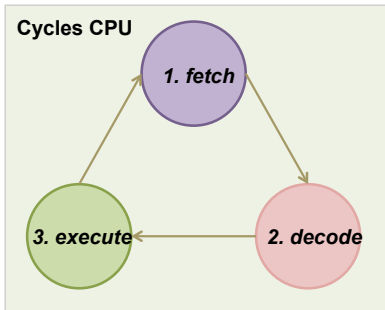
Principe de traitement de l'information

- Le traitement de données par un microprocesseur s'effectue en 3 cycles principaux

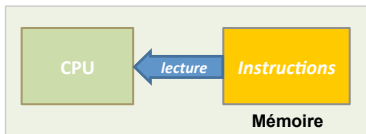
fetch lecture de l'instruction de la mémoire principale

decode décode de l'instruction à exécuter

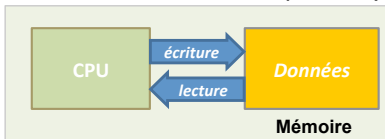
execute exécution de l'instruction et échange de données avec la mémoire principale



Chargement des instructions (*fetch*)



Traitement des données (*execute*)





- Avec l'évolution des ordinateurs, plusieurs types d'architectures ont été imaginées.
- En 1966, Michael J. Flynn propose de les classer en 4 catégories selon le flux de données et le flux d'instructions.
- Classification selon Flynn
 - ▶ Architecture SISD – Single Instruction Single Data
 - ▶ Architecture SIMD – Single Instruction Multiple Data
 - ▶ Architecture MISD – Multiple Instructions Single Data
 - ▶ Architecture MIMD – Multiple Instructions Multiple Data



■ Concept

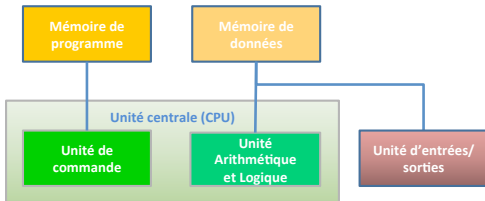
- ▶ Dans cette architecture, il n'existe aucun parallélisme, ni dans le flux de données, ni dans le flux d'instructions
- ▶ Chaque instruction traite une seule donnée à la fois
- ▶ Il s'agit d'un ordinateur séquentiel

■ Deux architectures fondamentales des microprocesseurs actuels

- ▶ Architecture Harvard (1944)
- ▶ Architecture Von Neumann (1945)



- Dans cette architecture, le programme et les données sont stockés dans des mémoires différentes et physiquement séparées
- Chacune de ces mémoires est accessible par un bus différent permettant ainsi d'augmenter la vitesse de traitement du microprocesseur (facteur 2)

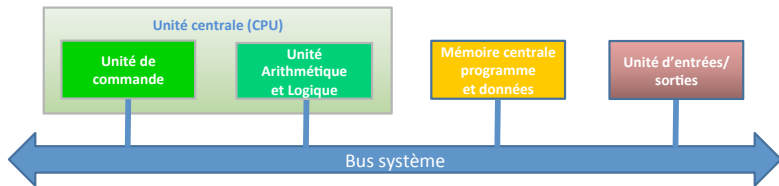


- Trop onéreux, il est « plus simple » de doubler la cadence d'accès à la mémoire
- Architecture abandonnée et reprise par les μP modernes pour l'implémentation des caches (instructions et données)



Architecture SISD - Von Neumann

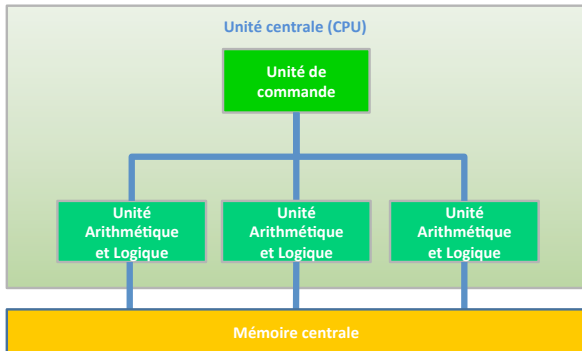
- Dans cette architecture, les données et le programme sont contenus dans une seule mémoire centrale
- L'unité d'entrées/sorties, la mémoire centrale et l'unité centrale sont interconnectées les unes aux autres par un seul bus système commun



- Tous les accès se font séquentiellement à travers un bus partagé entre les différentes unités et mémoires (*shared bus*)
- Le bus système est le facteur limitant de ce concept (au minimum 2 cycles pour lire une instruction et la donnée à traiter)

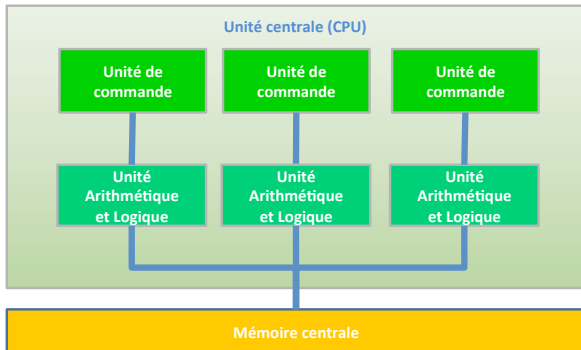


- L'unité centrale dispose de plusieurs ALU permettant un traitement simultané du flot de données
- Architecture largement utilisée par les processeurs graphiques (GPU)



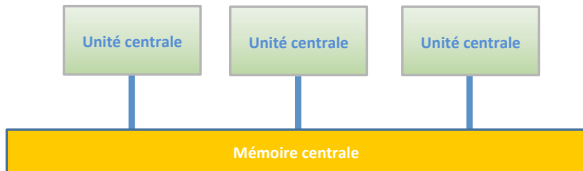


- L'unité centrale dispose de plusieurs unités de commandes et d'ALU permettant de traiter en parallèle la même donnée
- Il n'existe que très peu d'implémentations de cette architecture (IBM AP-101)





- Dans cette architecture, plusieurs unités centrales, appelées également « coeur » ou « core » en anglais, se partagent la même mémoire, mais chacune des unités exécute son propre programme
- Cette architecture permet de contourner les limitations de fréquence d'horloge liées à la physique et aux coûts des circuits intégrés
- Il est possible aujourd'hui d'intégrer plusieurs coeurs sur un même circuit





Performances d'un programme

- Les performances d'un programme peuvent être calculées comme suit

$$\frac{\text{time}}{\text{program}} = \frac{\text{time}}{\text{cycle}} * \frac{\text{cycles}}{\text{instruction}} * \frac{\text{instructions}}{\text{program}}$$

- Pour améliorer les performances d'un système à μP on peut agir sur
 - ▶ le temps par cycle
 - ▶ le nombre de cycles par instruction
 - ▶ le nombre d'instructions par programme



- Il existe deux grandes architectures pour les jeux d'instruction (ISA - Instruction Set Architecture)

- ▶ Architecture RISC

- ◇ Nombre réduit d'instructions utilisées très fréquemment, mais nécessitant l'utilisation d'un compilateur pour écrire les programmes
- ◇ Toutes les instructions ont la même taille
- ◇ Traitement d'une instruction en un seul cycle d'horloge

- ▶ Processeurs RISC

- ◇ ARM, PowerPC, MIPS, ...

- ▶ Architecture CISC

- ◇ Grand nombre d'instructions complexes pouvant être exécutées par le μP permettant d'écrire des programmes courts
- ◇ Taille des instructions de longueur variable
- ◇ Plusieurs cycles d'horloge nécessaire pour traiter une instruction

- ▶ Processeurs CISC

- ◇ x86, m68k, ...



- Sur les systèmes informatiques, il existe un très grand nombre de moyens pour stocker l'information et les programmes permettant de les traiter. Ces moyens vont fortement dépendre du type d'information à stocker et de leur volume.

- Dans le cadre des systèmes embarqués et des systèmes on chip, on distingue généralement deux types de mémoires
 - ▶ Mémoire vive
 - ▶ Mémoire permanente



Mémoire vive - RAM

- La mémoire vive est aussi appelée mémoire RAM (*Random Access Memory*)
- La mémoire vive est la mémoire de travail d'un μP . Elle lui permet de stocker les programmes et leurs données durant leur exécution.
- La mémoire vive est une mémoire volatile. Son contenu est effacé lors que le μP est éteint.
- Les deux grandes catégories de RAM sont
 - ▶ SRAM (*Static RAM*)

caractéristiques petite taille, très rapide, nécessite une alimentation pour conserver l'information

applications mémoire interne des μP , mémoire cache des μP

- ▶ DRAM (*Dynamic RAM*)

caractéristiques très grande taille, rapide, nécessite un rafraîchissement des cellules pour conserver l'information

applications mémoire principale externe des μP



Mémoire permanente - Flash

- La mémoire permanente est la mémoire des systèmes à μP servant au stockage des applications ainsi que des données persistantes (p. ex. données de configuration)
- La mémoire permanente est une mémoire non volatile. Son contenu est conservé lors que le μP est éteint.
- Les deux grandes technologies de flash sont
 - ▶ NOR-Flash

caractéristiques accès aléatoire à l'information, stockage garanti à 100%, taille réduite (MB), prix élevé

applications stockage de données de configuration, flash interne des μP

- ▶ NAND-Flash

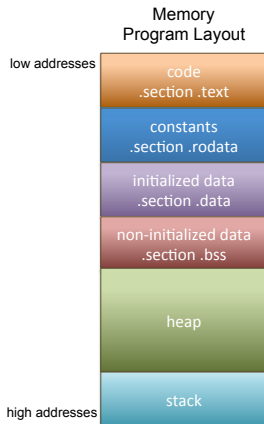
caractéristiques accès à l'information par page, stockage pas garanti à 100% (nécessite des mécanismes de correction), très grande taille (GB), prix bas

applications stockage des applications et données de configuration, flash externe des μP , disques (eMMC, SD-Card, SSD)



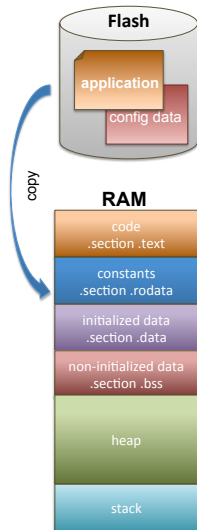
Mémoire - Layout d'un programme

- Un programme est structuré en plusieurs segments
- Chaque segment regroupe une partie distincte du logiciel et la stocke dans une section propre
 - ▶ le code (.section .text)
 - ▶ les constantes (.section .rodata)
 - ▶ les données initialisées (.section .data)
 - ▶ les données non-initialisées (.section .bss)
- En plus de ces quatre sections principales, un programme a besoin
 - ▶ d'une pile (stack) pour stocker les données locales et effectuer les appels de fonctions
 - ▶ d'un tas (heap) pour stocker les données dynamiques



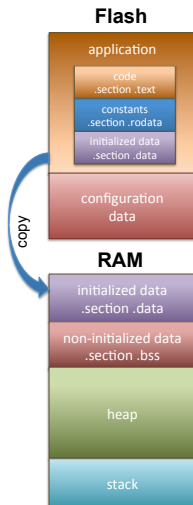


- Les systèmes embarqués disposent généralement de suffisamment de mémoire flash et RAM
- La mémoire flash est couramment organisée sous forme de systèmes de fichiers
- Les applications et les données de configuration sont stockées sous forme de fichiers
- Au lancement de l'application, celle-ci est chargée entièrement dans la mémoire vive (la RAM) pour être exécutée





- Les systèmes on chip disposent généralement que de très peu de mémoire flash et RAM
- La mémoire flash est normalement organisée sous forme de blocs de taille fixe
- Les applications et les données de configuration sont stockées directement dans les blocs de la mémoire flash
- Au lancement de l'application, seul le contenu de la section `.data` est chargé de la flash dans la RAM, le code est quant à lui exécuté directement depuis la flash





- La mémoire est vue comme un tableau à une dimension
- Chaque élément de la mémoire est adressable individuellement
- La taille minimale de ces éléments est de 8 bits, l'octet ou le byte
- En général et par convention, les positions mémoires croissent de haut en bas et la numérotation des bits se fait de droite à gauche
 - ▶ bit 0 : bit de poids faible
 - ▶ bit 7 : bit de poids fort
- La position dans le tableau sert d'adresse pour lire ou écrire un élément de mémoire

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
								H 0x0000
								E 0x0001
								I 0x0002
								A 0x0003
								- 0x0004
								F 0x0005
								R 0x0006
								0x0007
...								...
								0xfffe
								0xffff



- Certaines données nécessitent plus d'un octet pour être représentées, p. ex. les nombres entiers ont généralement une taille de 32 bits
- Il existe deux orientations possibles pour stocker une donnée en mémoire
 - ▶ **big-endian** (Motorola)
 - ◇ le byte de poids fort est stocké à l'adresse basse
 - ◇ le byte de poids faible est stocké à l'adresse haute
 - ◇ ce format est utilisé par les protocoles de communications TCP/IP
 - ▶ **little-endian** (Intel)
 - ◇ le byte de poids faible est stocké à l'adresse basse
 - ◇ le byte de poids fort est stocké à l'adresse haute



Mémoire - Endianness (II)

Little endian organization (Intel)

7	0
0x11	0
0x22	1
0x33	2
0x44	3
'a'	4
'b'	5
'c'	6
'd'	7

15	8	7	0
0x22	0x11		0
0x44	0x33		2
'b'	'a'		4
'd'	'c'		6
1	0		

31	24	23	16	15	8	7	0
0x44	0x33	0x22	0x11				0
'd'	'c'	'b'	'a'				4
3	2	1	0				

Donnée 32 bits : 0x44332211
String : "abcd"

Big endian organization (Motorola)

7	0
0x44	0
0x33	1
0x22	2
0x11	3
'a'	4
'b'	5
'c'	6
'd'	7

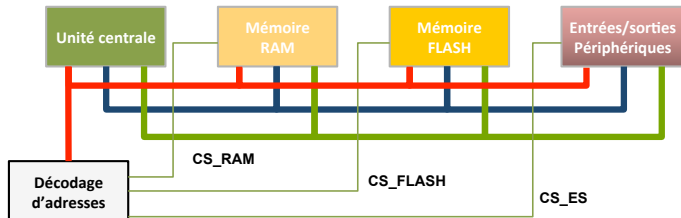
15	8	7	0
0x44	0x33		0
0x22	0x11		2
'a'	'b'		4
'c'	'd'		6
0	1		

31	24	23	16	15	8	7	0
0x44	0x33	0x22	0x11				0
'a'	'b'	'c'	'd'				4
0	1	2	3				



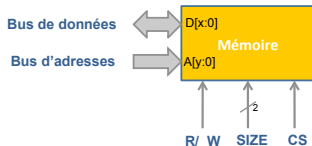
Bus microprocesseur

- Le microprocesseur communique avec la mémoire et les périphériques d'entrées/sorties au travers de lignes regroupées en bus
- Les signaux sont groupés selon leur fonctionnalité
 - ▶ Bus d'adresses [addr]
 - ▶ Bus de données [data]
 - ▶ Bus de contrôle [R/_W, CS, SIZE, ...]





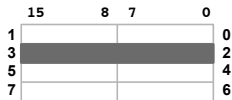
- Le bus de données permet d'échanger des informations (données ou instructions) entre l'unité centrale et la mémoire
- Les performances du système sont fortement influencées par la largeur du bus (8 bits, 16 bits ou 32 bits)



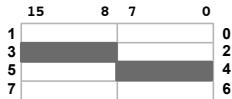
- Le bus d'adresses, piloté par le μP , transmet à la mémoire l'adresse de la donnée à échanger
- Le signal CS, piloté par le μP , indique à la mémoire que le transfert d'information va être effectué avec elle
- Le signal R/_W, piloté par le μP , indique à la mémoire s'il s'agit d'une opération de lecture ou d'écriture
- Les 2 signaux SIZE, pilotés par le μP , indiquent la taille de l'information échangée (8 bits, 16 bits ou 32 bits)



- Sur certains μP , le stockage d'information en mémoire peut être lié à des contraintes d'alignement
- L'accès aux données est limité à des adresses qui sont des multiples de la taille de la donnée en octets
- Pour qu'une donnée soit bien alignée, son adresse doit être divisible par sa taille en octets, p. ex.
 - ▶ 8 octets pour des mots de 64 bits
 - ▶ 4 octets pour des mots de 32 bits
 - ▶ 2 octets pour des mots de 16 bits
- Pour les μP supportant des données non alignées (missalignment), le non-alignement de certaines données influencera grandement les performances de l'application



aligned access



missaligned access