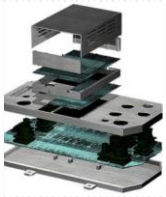


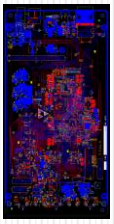

Architecture des systèmes embarqués

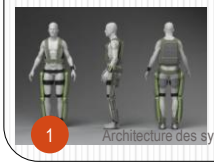


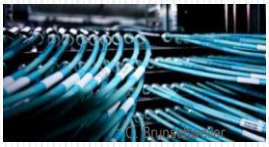
EISTI

2015 – 2016

Christophe Brunschweiler





Objectifs du cours

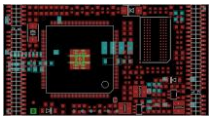
- Comprendre
  - Les spécificités des systèmes embarqués
  - Le fonctionnement intrinsèque des « ordinateurs »
  - L'organisation interne des « ordinateurs »
  - La représentation des données
  - L'exécution des programmes
- Connaître
  - Les différentes familles de micros utilisés dans l'embarqué
  - Certains protocoles intra cartes
- Etre capable de
  - Lire une datasheet de composant électronique (micro / périphérique)
  - Lire un schéma électronique d'un système embarqué
  - Mettre en œuvre un environnement de cross compilation (Eclipse / GCC)
  - Analyser un système embarqué réalisé à partir de composants COTS

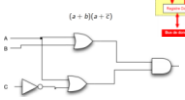
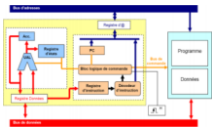
Déroulement du thème « ARCHI »

- 8 séances
  - 3h30 chacune
  - 13h30 – 17h00
  - De la théorie, mais pas uniquement…
- Du mardi 22 septembre au mardi 10 novembre
- Un examen final
  - Durée: 2 heures sur table
  - Documents interdits (en tout cas pour la première partie…)
  - Date: mercredi 2 décembre

Approche et progression

- Après une introduction générale aux systèmes embarqués et des rappels historiques et théoriques sur l'architecture des « ordinateurs », approche micro-macro des systèmes électroniques embarqués.

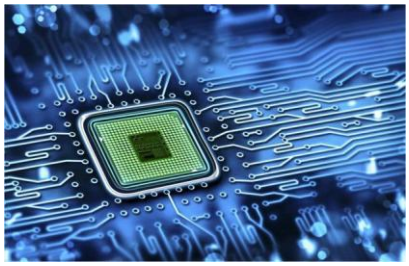




Organisation des 8 séances

- Séances 1, 2 et 3
  - Introduction
  - Historique des « ordinateurs »
  - Rappels théoriques
  - Diversité des « puces électroniques »
- Séances 4 et 5
  - Etude de quelques familles de micros (des plus basiques aux plus complexes)
- Séance 6
  - Mise en œuvre d'un STM32 à base d'ARM Cortex M4
  - Présentation de quelques protocoles intra-cartes
- Séance 7
  - Etude de quelques cartes électroniques et des interfaces avec le monde physique
  - Principes de conception des systèmes embarqués
- Séance 8
  - Cas concret: analyse de plusieurs systèmes embarqués

Plan



- Introduction
- Un peu d'histoire…
- Rappels théoriques
- Diversité des « puces électroniques »
- Familles de micros
- Mise en œuvre d'un STM32 à base d'ARM Cortex M4
- Conception des systèmes embarqués
- Du code au monde physique (par l'étude de quelques cartes – systèmes électroniques)
- Cas concrets: analyse de différents systèmes embarqués

Avant toute chose…

Créalie ESG open wide INGENIERIE

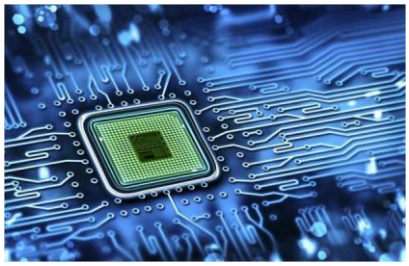
AGCO legrand Autoliv RENAULT BOSCH

somfy Axiohm Innovating for you ingenico DELPHI

ice Chubb UDRIVE European Naturalistic Driving Study Johnson Controls

hager VOLVO PSA PEUGEOT CITROËN Valeo

Plan



- **Introduction**
- Un peu d'histoire…
- Rappels théoriques
- Diversité des « puces électroniques »
- Familles de micros
- Mise en œuvre d'un STM32 à base d'ARM Cortex M4
- Conception des systèmes embarqués
- Du code au monde physique (par l'étude de quelques cartes – systèmes électroniques)
- Cas concrets: analyse de différents systèmes embarqués

Introduction

- Un système embarqué, c'est quoi?
- Principales caractéristiques des SE
- Principaux domaines d'application des SE
- Le marché des systèmes embarqués
- Le futur des systèmes embarqués

Systèmes / logiciels embarqués??
Embedded systems / software ??

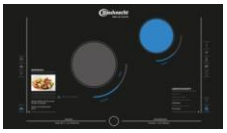
- « **Embedded** »: notion différente selon les personnes
  - Pour le « commun des mortels »
    - ?!?!?
  - Pour un développeur Web:
    - Un téléphone portable (même un smartphone…) est un système embarqué.
  - Pour un développeur de firmware sur cible 8 bits avec qq kB de ROM:
    - Tout ce qui dispose d'un OS n'est plus vraiment très « embedded »

Systèmes embarqués: de quoi parle t'on exactement?

- Un système embarqué n'est **pas** forcément:
  - mobile,
  - petit,
  - caché,
  - puissant (ou non puissant),
  - complexe,
  - alimenté par batterie,
  - économe en énergie,
  - cher (ou bon marché),
  - spécifique,
  - industriel,
  - …



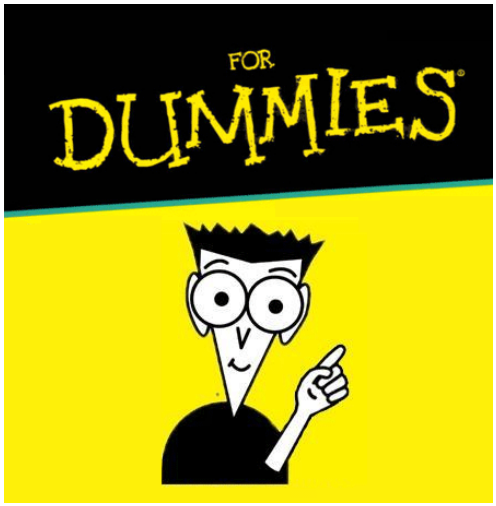


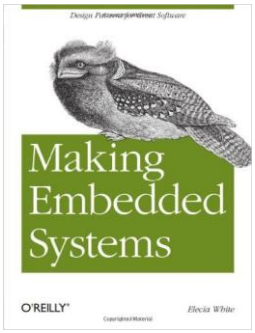

Définition « Embedded System »

- Tout système conçu **pour résoudre un problème ou une tâche spécifique / dédié** mais n'est pas un ordinateur d'usage général.
- Les **parties matérielle et logicielle sont intimement liées** et noyées dans le matériel et ne sont pas discernables comme un environnement de travail classique de type PC.
- Système électronique et informatique **autonome** possédant des **entrées-sorties spécifiques** (pour la plupart, liées à des grandeurs ou des phénomènes **physiques**).
- Composante primordiale d'un système plus large ou une machine dont l'objectif est de **commander, contrôler et superviser** ce système.

Embedded systems for dummies…



« Embedded systems are things like microwaves or auto-mobiles that runs software but aren't computers ».



Principales caractéristiques des SE

- Encombrement mémoire
- Consommation d'énergie
- Poids et volume
- Criticité / Fiabilité / Tolérance aux fautes
- Mobilité
- Communications
- Interfaçage avec monde physique
- Contraintes environnementales
- Contraintes temps réel
- Coûts
- MCO
- …

Principaux domaines d'application des SE

- Transports
  - Aéronautique
  - Automobile / Camions / Machinisme agricole
  - Ferroviaire
  - …
- Spatial
- Militaire / Défense
- Réseaux / Télécommunications
- GTB / Domotique / Bâtiments
- Distribution et gestion de l'énergie
- Médical
- Terminaux de paiement
- Consumer electronics
- Robotique
- …

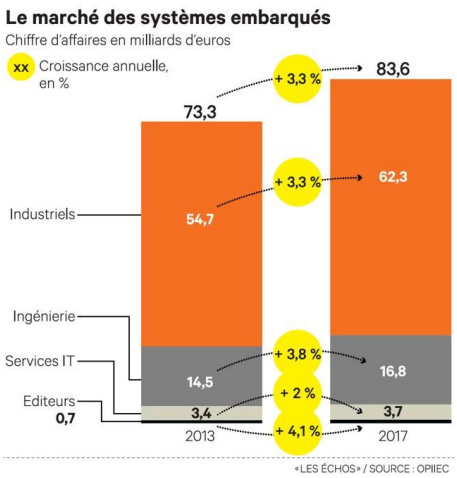
Le marché des systèmes embarqués (1/3)

- En France:
  - 76 000 entreprises (dont 13 000 purs spécialistes)
  - 387 500 emplois (220 000 en 2007)
  - CA de 73 Milliards € en 2013
  - 9% des offres d'emploi APEC
  - 41 000 créations de postes d'ici 2017

Source: les echos, 7/05/2015

- Selon IDC, le marché des systèmes embarqués "intelligents" dépassera les 1 000* milliards de dollars en 2019

Source: www.lembarque.com, 20/10/2014



**: 755 milliards $ actuellement*

Le marché des systèmes embarqués (2/3)

	Unit Chip Shipments (millions)	2012	2017	2020	2012-2020 Chip CAGR
Consumer/ Home	Mobile	4,800	9,100	10,800	12%
	Home	1,100	2,400	2,400	12%
	Home Networking	650	850	900	5%
Enterprise	Servers	40	50	65	7%
	Enterprise Networking	600	800	900	6%
	Storage	700	1,100	1,400	10%
Emb.	Internet-of-Things	760	2,000	3,000	22%
	Other Embedded	16,500	22,000	24,000	5%
	Other	2,300	3,300	4,300	9%
	Total	**27,000**	**41,000**	**48,000**	**9%**

Source: Gartner, IDC, SIA, and ARM estimates

Le marché des systèmes embarqués (3/3)

@DVANTAGE NOTE

So let us consider the highlights of the embedded system market

- $852bn Total embedded market 12% CAGR
- $113bn CPU embedded Market
- $3.31Bn packaged embedded software 7.1% CAGR ($6.5Bn in 2015)
- 10bn processors ship annually (98% embedded)
- 880k embedded software engineers globally 7% CAGR
- 40% of development projects behind schedule
- Shift to software as the critical and value creating element
- 50%+ of development costs relate to the software and its growing
- Professional growth is highest for development, test and QA engineers
- Total market for embedded s/w engineering $50bn (90% Labour $45bn, License & other $5Bn)

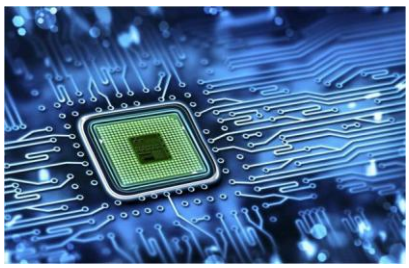
 Source: http://fr.slideshare.net/Declanka/note-16-embedded-advantage-1-0

SE: Technology drivers and enablers

- Toujours plus de **performances** (nombre d'opérations / quantité de données manipulées)
- Gagner en **autonomie** (= autonomous)
- Gagner en **consommation** énergétique
- Toujours plus de **connectivité**
- Améliorer la **réutilisation** et le **partage** (= bannir le propriétaire)
- Assurer la **sûreté** et la **sécurité**
- Toujours plus de **fiabilité**

 Source: http://fr.slideshare.net/Declanka/note-16-embedded-advantage-1-0

Plan



- Introduction
- **Un peu d'histoire…**
- Rappels théoriques
- Diversité des « puces électroniques »
- Familles de micros
- Mise en œuvre d'un STM32 à base d'ARM Cortex M4
- Conception des systèmes embarqués
- Du code au monde physique (par l'étude de quelques cartes – systèmes électroniques)
- Cas concrets: analyse de différents systèmes embarqués

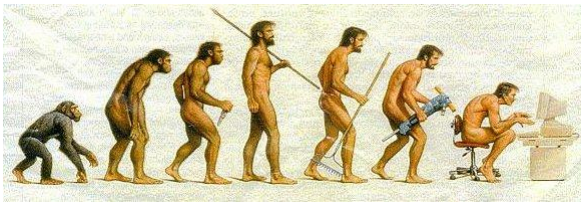
Un peu d'histoire…

D'où viennent les « ordinateurs »?

Quelques dates clés

Et du côté des systèmes embarqués…

D'où viennent les « ordinateurs »?



- L'ordinateur est né du besoin de calculer
  - Toujours plus complexe
  - Toujours plus vite

→ Automatisation des calculs

Quelques dates clés

Du boulier au premier transistor

- 500 av JC: Apparition des bouliers et abaques
- 1 632: Invention de la règle à calcul
- 1 642: Pascal invente la pascaline
- 1 833: Machine(s) à différence de Charles Babbage

Quelques dates clés

Du boulier au premier transistor

- 500 av JC: Apparition des bouliers et abaques
- 1 632: Invention de la règle à calcul
- **1 642: Pascal invente la pascaline**
- **1 833: Machine(s) à différence de Charles Babbage**

Une Pascaline permet d'additionner et de soustraire deux nombres d'une façon directe et de faire des multiplications et des divisions par répétitions.

https://fr.wikipedia.org/wiki/Pascaline

Le but de la machine à différence de Babbage est de calculer les polynômes en utilisant une méthode de calcul dite méthode des différences.

https://fr.wikipedia.org/wiki/Charles_Babbage

Quelques dates clés

Du boulier au premier transistor

- 500 av JC: Apparition des bouliers et abaques
- 1 632: Invention de la règle à calcul
- 1 642: Pascal invente la pascaline
- 1 833: Machine(s) à différence de Charles Babbage
- 1 854: Publication par Georges Boole d'un ouvrage sur la logique (algèbre booléenne)
- 1 904: Invention du tube à vide

Quelques dates clés

Du boulier au premier transistor

- 500 av JC: Apparition des bouliers et abaques
- 1 632: Invention de la règle à calcul
- 1 642: Pascal invente la pascaline
- 1 833: Machine(s) à différence de Charles Babbage
- 1 854: Publication par Georges Boole d'un ouvrage sur la logique (algèbre booléenne)
- **1 904: Invention du tube à vide**

Tube à vide (= tube électronique ou lampe)

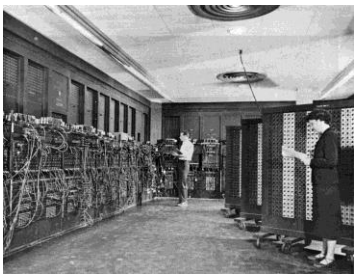
Quelques dates clés

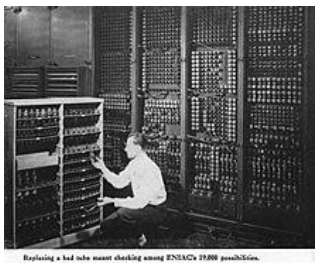
Du boulier au premier transistor

- 500 av JC: Apparition des bouliers et abaques
- 1 632: Invention de la règle à calcul
- 1 642: Pascal invente la pascaline
- 1 833: Machine(s) à différence de Charles Babbage
- 1 854: Publication par Georges Boole d'un ouvrage sur la logique (algèbre booléenne)
- 1 904: Invention du tube à vide
- 1 937: Article d'Alan Turing sur la calculabilité: machines de Turing
- 1 943: Création du ASCC Mark 1 (Harvard - IBM) : Automatic Sequence-Controlled Calculator
- 1 946: Construction de l'ENIAC

ENIAC: Electronic Numerical Integrator Analyser and Computer

- 17 468 tubes à vide
- 7 200 diodes à cristal
- 1 500 relais
- 70 000 résistances
- 10 000 condensateurs
- 5 millions de soudures faites à la main
- 30 tonnes
- 167 mètres carrés
- Consommation 150 kilowatts
- Puissance de calcul*:
  - 1 ms pour multiplier 2 nombres de 10 chiffres
  - 3 secondes pour calculer une trajectoire de tir





Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

Quelques dates clés

Du boulier au premier transistor

- 500 av JC: Apparition des bouliers et abaques
- 1 632: Invention de la règle à calcul
- 1 642: Pascal invente la pascaline
- 1 833: Machine(s) à différence de Charles Babbage
- 1 854: Publication par Georges Boole d'un ouvrage sur la logique (algèbre booléenne)
- 1 904: Invention du tube à vide
- 1 937: Article d'Alan Turing sur la calculabilité: machines de Turing
- 1 943: Création du ASCC Mark 1 (Harvard - IBM) : Automatic Sequence-Controlled Calculator
- 1 946: Construction de l'ENIAC
- 1 947: Invention du transistor (Bell)

Quelques dates clés

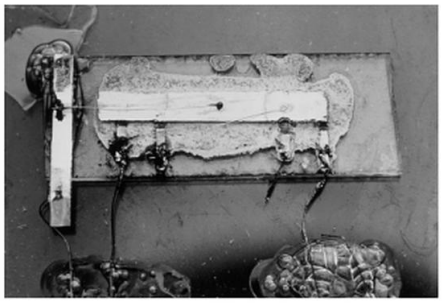
Du premier ordinateur à l'IBM PC et MS-DOS…

- 1 955: Premier ordinateur à transistors (TRADIC – Bell)
- 1 958: Premier circuit intégré (Texas Instrument)

Quelques dates clés

Du premier ordinateur à l'IBM PC et MS-DOS…

- 1 955: Premier ordinateur à transistors (TRADIC – Bell)
- **1 958: Premier circuit intégré (Texas Instrument)**



Le premier circuit intégré chez Texas (1958) (Texas Instruments Incorporate).

Quelques dates clés

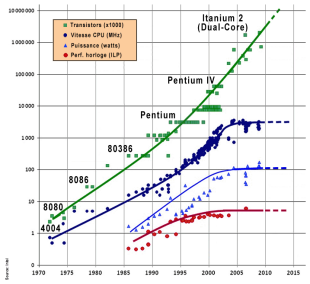
Du premier ordinateur à l'IBM PC et MS-DOS…

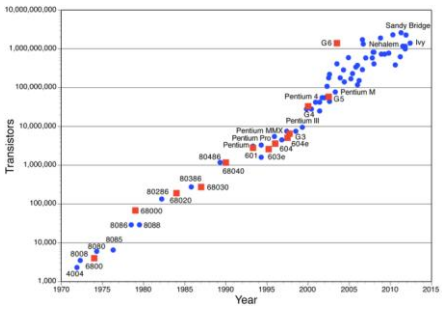
- 1 955: Premier ordinateur à transistors (TRADIC – Bell)
- 1 958: Premier circuit intégré (Texas Instrument)
- 1 965: G. Moore énonce la loi qui porte son nom (loi de Moore)

Quelques dates clés

Du premier ordinateur à l'IBM PC et MS-DOS…

- 1 955: Premier ordinateur à transistors (TRADIC – Bell)
- 1 958: Premier circuit intégré (Texas Instrument)
- **1 965: G. Moore énonce la loi qui porte son nom (loi de Moore)**





Quelques dates clés

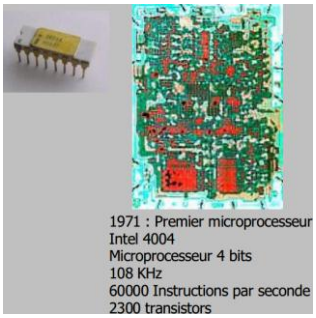
Du premier ordinateur à l'IBM PC et MS-DOS…

- 1 955: Premier ordinateur à transistors (TRADIC – Bell)
- 1 958: Premier circuit intégré (Texas Instrument)
- 1 965: G. Moore énonce la loi qui porte son nom (loi de Moore)
- 1 969: Système d'exploitation MULTICS puis UNIX (Bell)
- 1 971 / 1972: 1er microprocesseur, le 4004 d'Intel puis le 8008

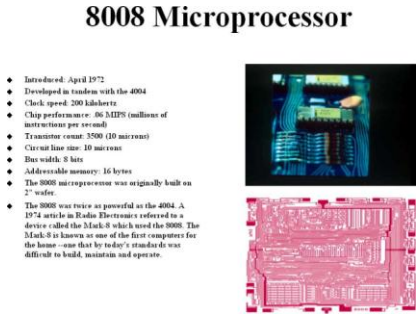
Quelques dates clés

Du premier ordinateur à l'IBM PC et MS-DOS…

- 1 955: Premier ordinateur à transistors (TRADIC – Bell)
- 1 958: Premier circuit intégré (Texas Instrument)
- 1 965: G. Moore énonce la loi qui porte son nom (loi de Moore)
- 1 969: Système d'exploitation MULTICS puis UNIX (Bell)
- **1 971 / 1972: 1er microprocesseur, le 4004 d'Intel puis le 8008**



1971 : Premier microprocesseur
Intel 4004
Microprocesseur 4 bits
108 KHz
60000 Instructions par seconde
2300 transistors



Quelques dates clés

Du premier ordinateur à l'IBM PC et MS-DOS…

- 1 955: Premier ordinateur à transistors (TRADIC – Bell)
- 1 958: Premier circuit intégré (Texas Instrument)
- 1 965: G. Moore énonce la loi qui porte son nom (loi de Moore)
- 1 969: Système d'exploitation MULTICS puis UNIX (Bell)
- 1 971 / 1972: 1er microprocesseur, le 4004 d'Intel puis le 8008
- 1 972: Micro-ordinateur Micral (R2E, rachetée ensuite par Bull)
- 1 973: Langage C pour le développement d'Unix
- 1 974: Premier microprocesseur Motorola: 6800
- 1 975: Micro-ordinateur Altair 8800 (MITS), à base d'Intel 8080

Altair 8800 de MITS

- Constructeur : MITS
- Model : Altair 8800
- Date de lancement : janvier 1975
- Pays d'origine : États-Unis
- Prix : 400 dollars en kit, 600 dollars monté
- Processeur Intel 8080 et 8 080 A
- Vitesse : 2 MHz
- Mémoire vive : 256 octets extensible à 64 ko
- ROM : optionnelle
- Sauvegarde optionnelle : lecteur de bande perforée, de cassette, ou disquette 5.25 pouces
- Extension : 4 slots de carte mère extensible à 16 slots total
- Bus : S-100
- Vidéo : jeux de diodes électroluminescentes sur la face avant
- Clavier : jeux d'interrupteurs sur la face avant
- Port I/O : série et parallèle
- Système d'exploitation : CP/M, MS-DOS, Altair Disk BASIC

https://fr.wikipedia.org/wiki/Altair_8800

Quelques dates clés

Du premier ordinateur à l'IBM PC et MS-DOS…

- 1 955: Premier ordinateur à transistors (TRADIC – Bell)
- 1 958: Premier circuit intégré (Texas Instrument)
- 1 965: G. Moore énonce la loi qui porte son nom (loi de Moore)
- 1 969: Système d'exploitation MULTICS puis UNIX (Bell)
- 1 971 / 1972: 1er microprocesseur, le 4004 d'Intel puis le 8008
- 1 972: Micro-ordinateur Micral (R2E, rachetée ensuite par Bull)
- 1 973: Langage C pour le développement d'Unix
- 1 974: Premier microprocesseur Motorola: 6800
- 1 975: Micro-ordinateur Altair 8800 (MITS), à base d'Intel 8080
- 1 976 / 1 977: Apple I (200 p) puis Apple II (produit jusqu'en 1988)

Apple I et Apple II

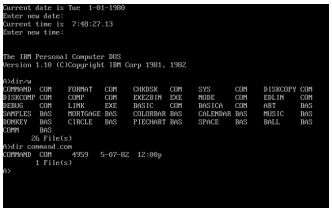
Quelques dates clés

Du premier ordinateur à l'IBM PC et MS-DOS…

- 1 955: Premier ordinateur à transistors (TRADIC – Bell)
- 1 958: Premier circuit intégré (Texas Instrument)
- 1 965: G. Moore énonce la loi qui porte son nom (loi de Moore)
- 1 969: Système d'exploitation MULTICS puis UNIX (Bell)
- 1 971 / 1972: 1er microprocesseur, le 4004 d'Intel puis le 8008
- 1 972: Micro-ordinateur Micral (R2E, rachetée ensuite par Bull)
- 1 973: Langage C pour le développement d'Unix
- 1 974: Premier microprocesseur Motorola: 6800
- 1 975: Micro-ordinateur Altair 8800 (MITS), à base d'Intel 8080
- 1 976 / 1 977: Apple I (200 p) puis Apple II (produit jusqu'en 1988)
- 1 981: IBM PC et MS-DOS

IBM PC et MS-DOS

- Date de sortie: 1981
- Fin de production: 1985
- Environnement:
  - lecteurs de disquettes 5 pouces 1/4 de 160 ko
  - ports ISA
- Système d'exploitation: PC-DOS / MS-DOS
- Processeur: Intel 8088
- Mémoire: 16 ko extensible à 256 ko
- Écran monochrome en 25 lignes de 80 caractères
- Résolution: 320x200

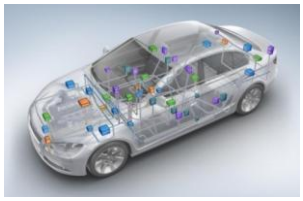


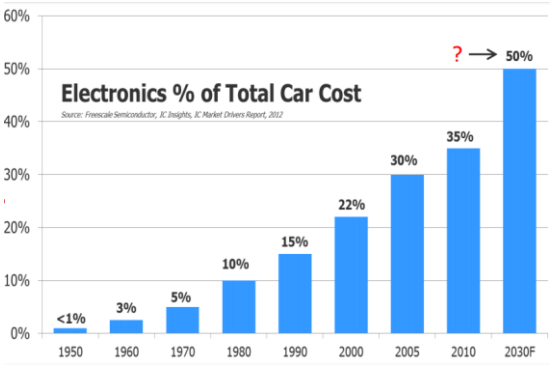
Quelques dates clés

Des protocoles TCP/IP au milliard de sites Web…

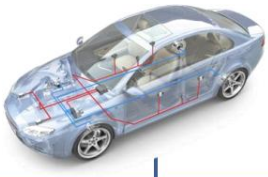
- 1 982: Définition du protocole TCP/IP et du mot Internet
- 1 983: Création du C++
- 1 984: Sortie du Macintosh et de Mac OS
- 1 989: Invention du World Wide Web
- 1 991: Premier noyau Linux 0.01
- 1 993: Mosaic, le premier navigateur web
- 1 995: Création du langage de programmation Java
- 1 996: Première version de la norme USB
- 2 000: Premier Baladeur MP3 à disque dur (Archos)
- 2 003: le nombre d'utilisateurs Linux est estimé à 18 millions
- 2 007: Premier Iphone
- 2 009: Première tablette sous Android
- 2 011: Les ventes de smartphones dépassent celles de PC
- 2 014: Le nombre de sites web dans le monde dépasse le milliard

Et du côté des systèmes embarqués…





Vehicle domaines and main components



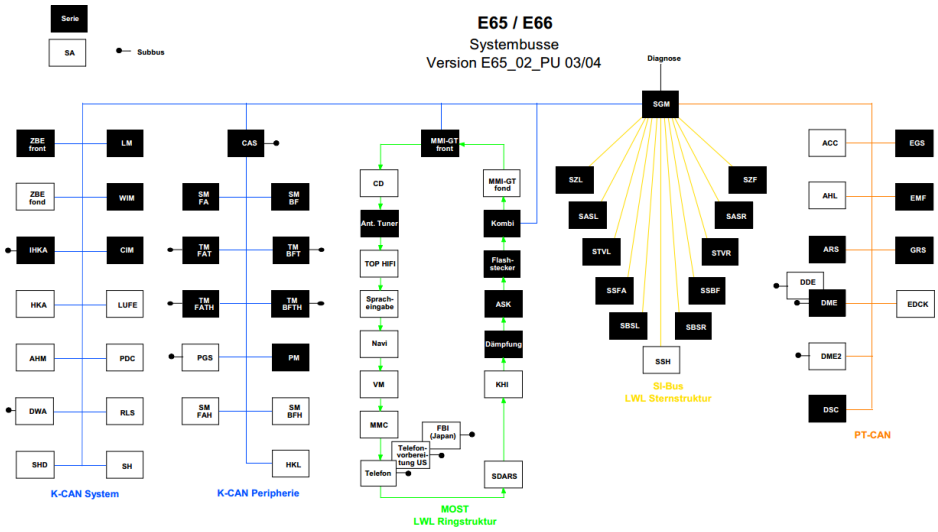
Source: Bosch

Evolution des fonctions électroniques dans l'automobile

	1960	1970	1980	1990	2000	2010
Drivetrain	• Ignition		• Fuel Injection • Enginecontrol Otto	• Valve control • Diesel pump • Slip control	• FSI • Pumpe-Düse-ECU • 32bit Controller	• electromagn. Valves? • Hybrid • Fuel cell •
Chassis			• ABS • controlled Damping	• ESP • Bremsassistent	• elektrohydr. Brake • brake-by-wire? • Autom. Cruise Control • autom.emerg. stop • ACC Stop+go • UVF? • Lenkhilfe • Überlagerungslenkung • steer-by-wire? • skyhook-control • Wankausgleich	
Safety			• Airbag		• 2step Airbags • Pedestrian Protect. • byteflight • precrash	
Comfort	• intervall Wiper		• Climate control	• Xenon-lights	• Keyless Entry• advanced frontlighting • 2Motor-Wiper	
Power+Wirung	• 12V		• CAN	• D2B .watercooled.Generator • • elektron. ZE	• MOST,LIN • TTP/Flexray • Startergenerator • APU? • power module • 42V?.	
Information	• Radio		• Trip computer	• Sound systems Satellite radio• • GSM • GPS Navigation	• TV • Infotainment • DAB • bluetooth • UMTS • Internet • Veh.-Veh.-Comm.	

Source: UNIKASSEL VERSITÄT

Exemple d'architecture EE d'un véhicule



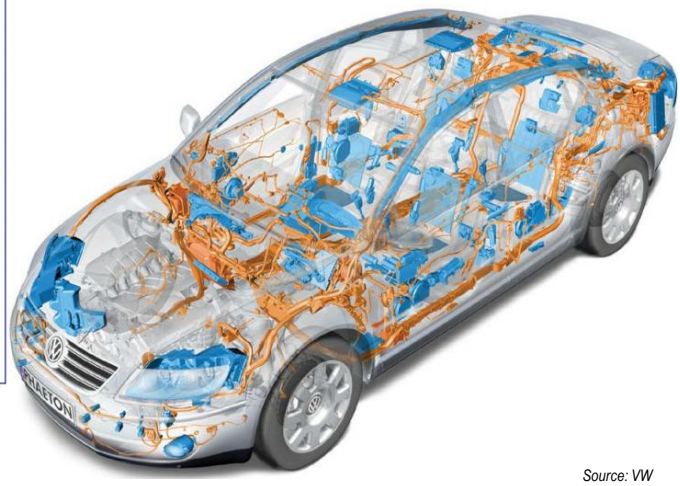
Source: BMW

Une électronique omni présente…

VW Phaeton:
- 11.136 electrical parts in total

communication:
- **61 ECUs in total**
- external diagnosis for 31 ECUs via serial communication
- optical bus for high bandwidth Infotainment-data
- **sub-networks** based on proprietary serial bus
- **35 ECUs** connected by **3 CAN-busses**

sharing
- appr. **2500 signals**
- in **250 CAN messages**



Source: VW

Un véhicule plein de superlatifs

Example: 2014 Mercedes S Class


THE HANSEN REPORT ON AUTOMOTIVE ELECTRONICS

Mercedes...

- Maximum network ECU connections: 10 for FlexRay, 73 for CAN and 61 for LIN
- Approximately 200 microprocessors
- Up to 65 million lines of code, 30 million for the multimedia system alone
- Four network gateways: EES, the emissions ECU; CPC, the central powertrain controller; the telematics head unit; and ESP, the electronics stability program ECU
- Base vehicles employ 1,376 wires with a total length of 2,474 meters. A fully optioned vehicle requires 2,385 wires, with a total length of 4,293 meters.
- 500 LEDs are deployed, no light bulbs
- 100 motors in the interior
- 156 buttons and switches in the interior
- The price for the S 350 BlueTEC short wheelbase diesel S-Class starts at €79,789.50.
- Equipped with a gasoline engine, the long wheelbase S 500 starts at €107,635.50.

Maximum network ECU connections: 10 for FlexRay, 73 for CAN and 61 for LIN

Approximately 200 microprocessors

Up to 65 million lines of code, 30 million for the multimedia system alone

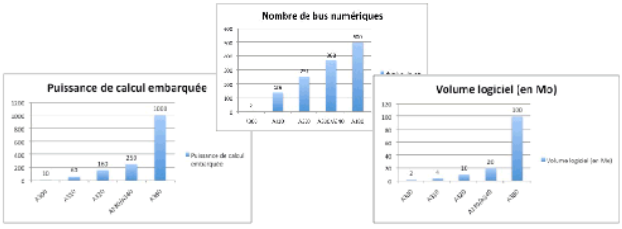
Base vehicles employ 1,376 wires with a total length of 2,474 meters. A fully optioned vehicle requires 2,385 wires, with a total length of 4,293 meters.

500 LEDs are deployed, no light bulbs

100 motors in the interior

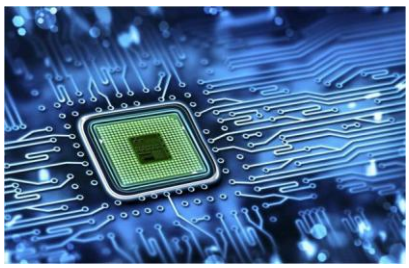
Source: The Hansen Report On Automotive Electronics, Jul 2013

Et dans l'aéronautique?



- L'avionique des Airbus A310 en 1980 représentait 4 Mo de codes, s'appuyant sur 77 calculateurs et 136 bus numériques.
- En 1990, sur l'A340, le code faisait 20 Mo, s'appuyant sur 115 calculateurs et 368 bus numériques.
- Les systèmes embarqués de l'A380 comprennent 100 millions de lignes de codes

Plan



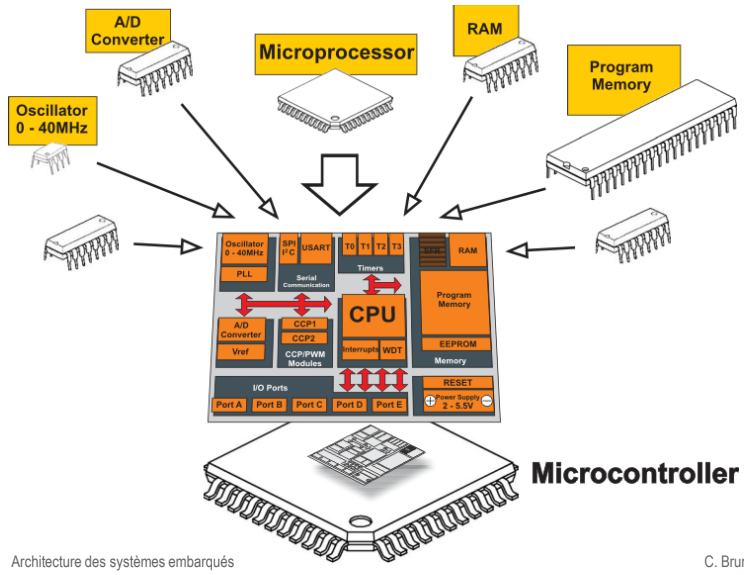
- Introduction
- Un peu d'histoire…
- **Rappels théoriques**
- Diversité des « puces électroniques »
- Familles de micros
- Mise en œuvre d'un STM32 à base d'ARM Cortex M4
- Conception des systèmes embarqués
- Du code au monde physique (par l'étude de quelques cartes – systèmes électroniques)
- Cas concrets: analyse de différents systèmes embarqués

Rappels théoriques

La représentation des nombres
La représentation des nombres en mémoire
Traitements mathématiques (ex: l'additionneur)
L'unité de traitement
L'architecture de Von Neuman
L'architecture de Harvard
Le traitement des instructions
Notion de jeu d'instructions
Assembleur et langage machine
Les jeux d'instruction RISC vs CISC
Le pipelining
Mémoires au pluriel… (ROM, RAM, Registres, Cache, Modes d'adressage, MMU, la pile, …)
Les interruptions

Préambule

Microprocesseur vs Microcontrôleur



La représentation des nombres

Système binaire

- Toutes les informations sont représentées sous forme **binaire**
- Le **codage des informations** est la fonction établissant une correspondance entre la représentation externe de l'information (e.g., A, 36, une image, un son) et sa représentation interne qui est une suite de bits (e.g., 01000001, 100100, ...)
- Le **système binaire** est le système de numérotation utilisant la **base 2**.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	0	0	1	1	0	0

- Un « mot » de n bits permet de représenter **2^n possibilités**

La représentation des nombres

Codage des données numériques - Entiers positifs ou nuls

Tout nombre entier positif (de n chiffres a_i) peut être représenté, en base b, par une expression de la forme :

$$x = a_{n-1} * b^{n-1} + a_{n-2} * b^{n-2} + ... + a_1 * b^1 + a_0 * b^0$$

Additionner les puissances de 2 correspondants aux bits de valeur 1.

$$101001111 = 2^8 + 2^6 + 2^3 + 2^2 + 2^1 + 2^0 = 256 + 64 + 8 + 4 + 2 + 1 = 335$$

Attention au poids !

- **bit de poids faible** : le bit ayant la moindre valeur (i.e., celui de droite)
- **bit de poids fort** : le bit ayant la plus grande valeur (i.e., celui de gauche)

La représentation des nombres

Codage des données numériques - Entiers négatifs

- Plusieurs façons de représenter un nombre négatif
  - Valeur absolue signée
  - Complément à 1
  - Complément à 2
- Exemples (codage du nombre -6 = 0110):
  - Valeur absolue signée: 1110
  - Complément à 1: 1001
  - Complément à 2: 1010
- Dans les ordinateurs, c'est la représentation en **complément à 2** qui a été choisie.

La représentation des nombres

Codage des données numériques - Nombres fractionnaires

- Représentation de la virgule:
  - **Virgule fixe**: nombre fixe de chiffres après la virgule

Exemple : $(25,75)_{10} = (11001,110)_2$

Sur 10 bits (7 partie entière, 3 partie décimale binaire) :

0	0	1	1	0	0	1	1	1	0

  - **Virgule flottante**: la position de la virgule n'est pas fixe.

→ Voir norme IEEE 754

La représentation des nombres en mémoire
Little endian / big endian

Little/Big Endian

- Petit/Gros boutiste (étymologie : voyage de Gulliver)
- exemple : la valeur 433=0x01B1 nécessitant 2 octets en mémoire
- Big endian : most significant bit (MSB : 01h) en premier
- Little Endian : lest significant bit (LSB : B1h) en premier

Little-Endian (x86,arm)

adresse mémoire	00	01
	B1	01

Big-Endian (powerpc,arm,mips)

adresse mémoire	00	01
	01	B1

- Protocole IP : big endian (pour tous, interopérabilité)

Source: Loïc Cuvillon

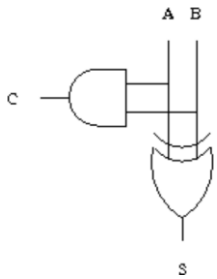
Traitements mathématiques
Additionneur

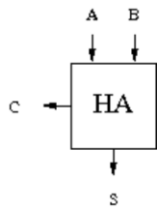
- Semi-additionneur

Entrées		Sorties	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = A \oplus B$$

$$C = A \wedge B$$



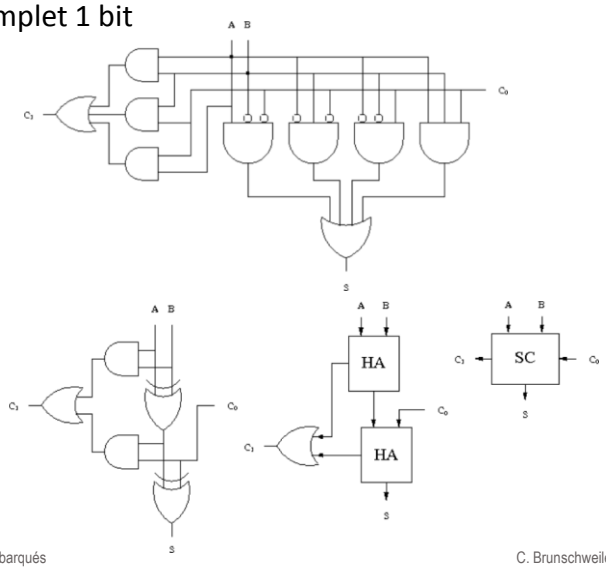


Traitements mathématiques

Additionneur

- Additionneur complet 1 bit

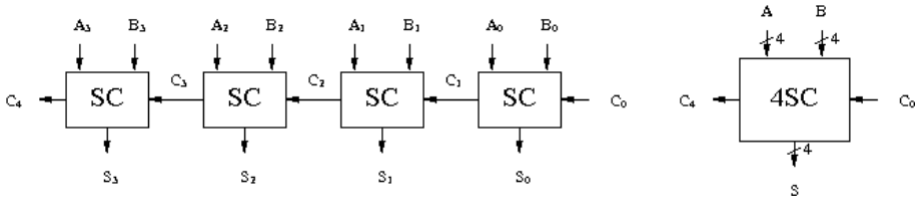
Entrées			Sorties	
A	B	C_0	C_1	S
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1



Traitements mathématiques

Additionneur

- Additionneur à propagation de retenue



Additionneur 4 bits

Traitements mathématiques

Calcul des indicateurs

Indicateur N (pour Négatif)

Il indique si le résultat est négatif. Comme les entiers sont représentés en compléments à 2, il est égal au bit de poids fort du résultat.

Indicateur Z (pour Zéro)

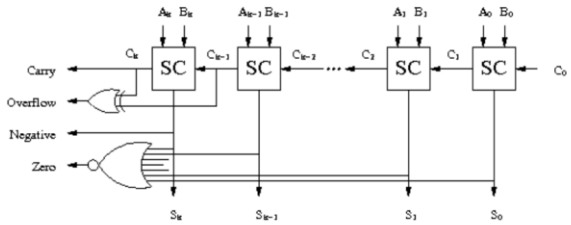
Il indique si le résultat est égal 0. Il est donc égal au complémentaire du *ou logique* (c'est-à-dire un *nor*) de tous les bits du résultat.

Indicateur C (pour Carry)

Il indique si l'opération a provoqué une retenue. Il est peut donc uniquement être positionné par les opérations arithmétiques. Il est mis à 1 lorsqu'il y a une retenue. Ceci correspond à un débordement pour une addition de nombres non signés.

Indicateur O (pour Overflow)

Il indique un débordement lors d'une addition de nombres signés. Ce débordement intervient lorsque la somme de deux nombres positifs est supérieure à 2^{k-1} ou lorsque la somme de deux nombres négatifs est inférieure à $-2^{k-1}-1$. Cet indicateur est égal au *ou exclusif* $C_k \oplus C_{k-1}$ des deux dernières retenues.



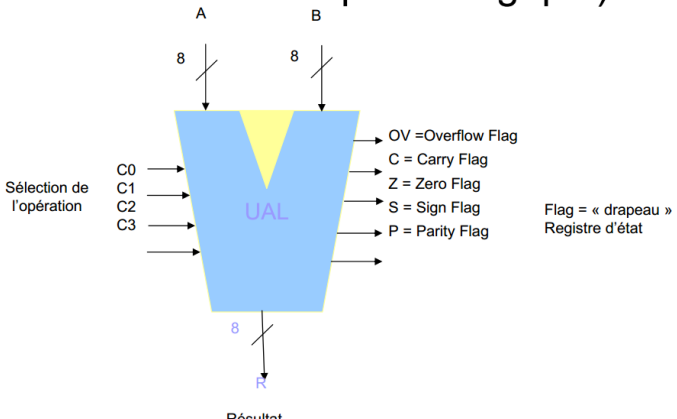
Calcul des indicateurs

Source: Olivier Carton

L'unité de traitement

- ALU ou UAL: regroupe l'ensemble des traitements pouvant être appliqués à des données

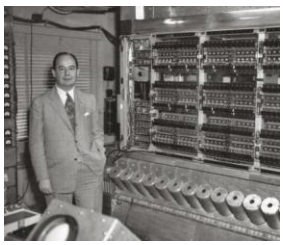
UAL : Unité Arithmétique et Logique)

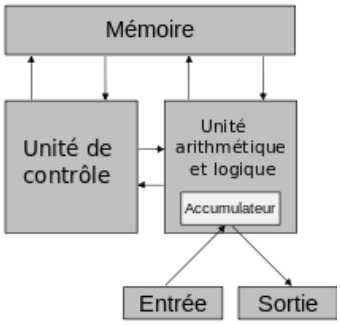


Source: Sylvain MONTAGNY

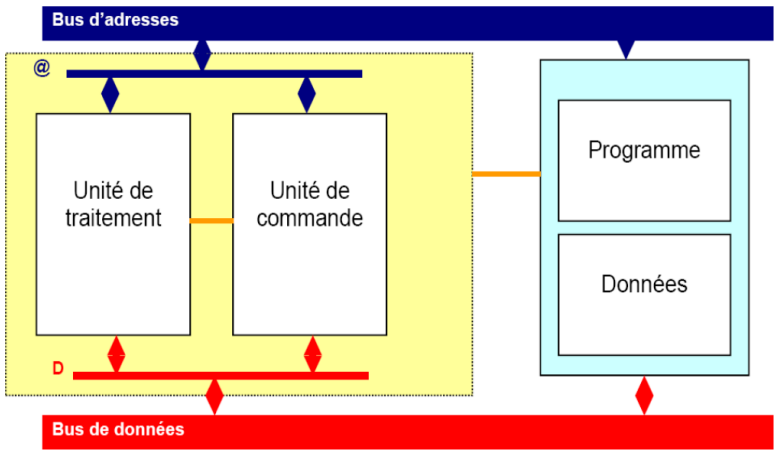
L'architecture de Von Neumann

- Du nom de John von Neumann, ayant participé à la construction de l'ENIAC.
- Proposition, en 1945 d'une architecture à base de 4 sous ensembles:
  - **l'unité arithmétique et logique** (UAL) ou unité de traitement, qui effectue les opérations de base ;
  - **l'unité de contrôle**, qui est chargée du séquençage des opérations ;
  - **la mémoire**, qui contient à la fois les données et le programme qui indique à l'unité de contrôle quels calculs faire sur ces données. La mémoire se divise en mémoire vive (programmes et données en cours de fonctionnement) et mémoire de masse (programmes et données de base de la machine) ;
  - **les dispositifs d'entrées-sorties**, qui permettent de communiquer avec le monde extérieur.
- C'est l'architecture la plus utilisée dans les micro-processeurs (contrôleurs)!



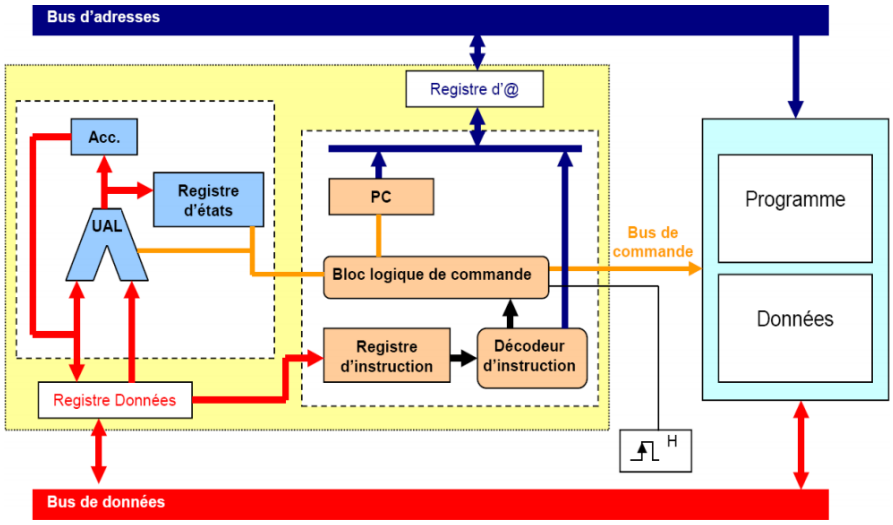


L'architecture de Von Neumann



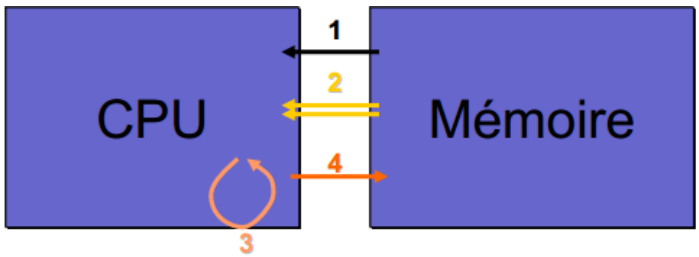
Source: Sylvain MONTAGNY

L'architecture de Von Neumann



Source: Sylvain MONTAGNY

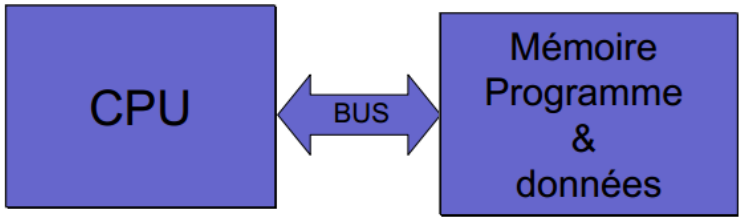
Fonctionnement basique d'une opération de calcul



- (1) Charger une instruction depuis la mémoire
- (2) Charger les opérandes depuis la mémoire
- (3) Effectuer les calculs
- (4) Stocker le résultat en mémoire

Source: Sylvain MONTAGNY

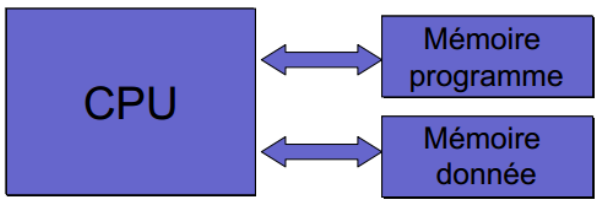
L'architecture
Von Neuman



- Un seul chemin d'accès à la mémoire
  - Un bus de données (programme et données),
  - Un bus d'adresse (programme et données)
- Architecture des processeurs d'usage général
- Goulot d'étranglement pour l'accès à la mémoire

Source: Sylvain MONTAGNY

L'architecture
Harvard



- Séparation des mémoires programme et données
  - Un bus de données programme,
  - Un bus de données pour les données,
  - Un bus d'adresse programme,
  - Un bus d'adresse pour les données.
- Meilleure utilisation du CPU :
  - Chargement du programme et des données en parallèle

Source: Sylvain MONTAGNY

Le traitement des instructions
Exemple d'instruction

- Instruction Addition :

  Accumulateur = Accumulateur + Opérande

Correspond à l'instruction ADD A,#2

Instruction (16 bits)	
Code opératoire (5 bits)	Champ opérande (11 bits)
ADD A	#2
11001	000 0000 0010

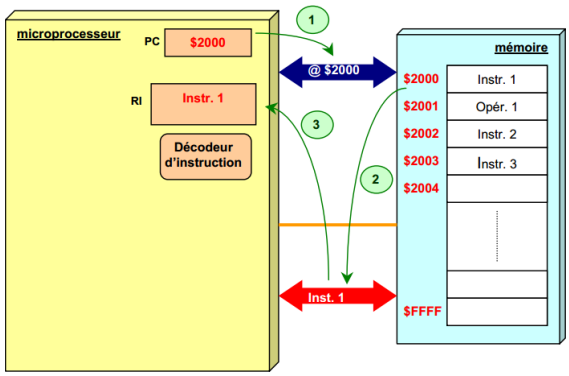
Cette instruction est comprise par le processeur par le mot binaire :

11001 000 0000 0010 = code machine

Source: Sylvain MONTAGNY

Le traitement des instructions
Phase 1 : Recherche de l'instruction en mémoire

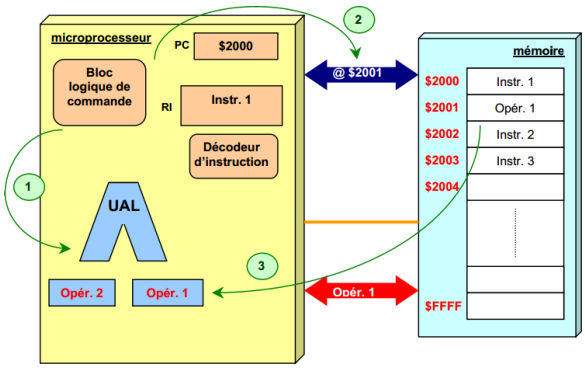
- La valeur du PC est placée sur le bus d'adresse par l'unité de commande qui émet un ordre de lecture.
- Après le temps d'accès à la mémoire, le contenu de la case mémoire sélectionnée est disponible sur le bus des données.
- L'instruction est stockée dans le registre d'instruction du processeur.



Source: Sylvain MONTAGNY & T. Dumartin

Le traitement des instructions
Phase 2 : Décodage et recherche de l'opérande

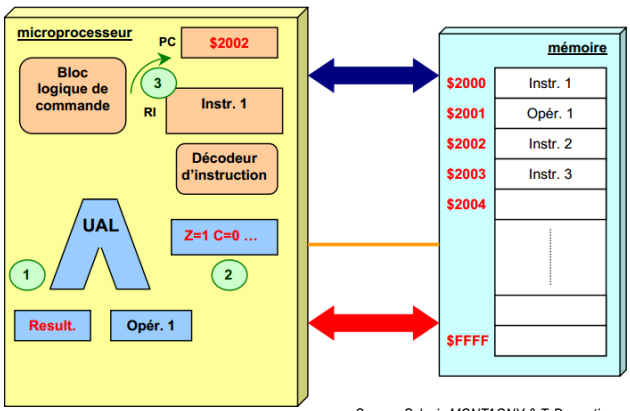
- L'unité de commande transforme l'instruction en une suite de commandes élémentaires nécessaires au traitement de l'instruction.
- Si l'instruction nécessite une donnée en provenance de la mémoire, l'unité de commande récupère sa valeur sur le bus de données.
- L'opérande est stocké dans le registre de données.



Source: Sylvain MONTAGNY & T. Dumartin

Le traitement des instructions
Phase 3 : Exécution de l'instruction

- Le séquenceur réalise l'instruction.
- Les drapeaux sont positionnés (registre d'état).
- L'unité de commande positionne le PC pour l'instruction suivante.



Source: Sylvain MONTAGNY & T. Dumartin

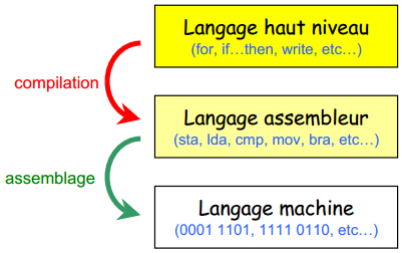
Notion de **jeu d'instructions**

- Le jeu d'instructions décrit l'ensemble des opérations élémentaires que le processeur peut exécuter.
- Les instructions peuvent être classées en 4 groupes
  - **Transfert de données** (charger ou sauver en mémoire, etc…)
  - **Opérations arithmétiques** (addition, soustraction, etc…)
  - **Opérations logiques** (ET, OU, comparaison, etc…)
  - **Contrôle de séquence** (branchements, tests, etc…)
- Chaque instruction peut être associée à un ou plusieurs **modes d'adressage,** par exemple:
  - Adressage de registre
  - Adressage immédiat
  - Adressage direct
- Chaque instruction nécessite un certain nombre de **cycles d'horloges** pour s'effectuer.

Source: Sylvain MONTAGNY

Assembleur et langage machine

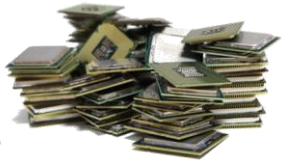
- Le langage assembleur est le plus proche du langage machine.
- Chaque processeur (famille de processeurs) possède son propre langage assembleur.
- Exemple (ARM):




```
MOV     r3,#15     @ r3 <- 0x0000000F
MOV     r4,#0x2E   @ r4 <- 0x0000002E
MOV     r0,r3      @ r0 <- r3 = 0x0000000F
MOV     r1,r4      @ r1 <- r4 = 0x0000002E
```


Source: Sylvain MONTAGNY

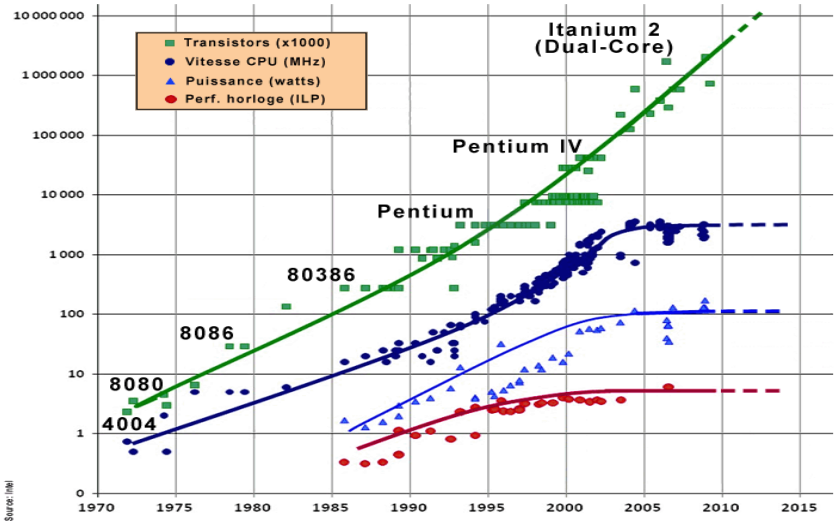
Performance d'un processeur



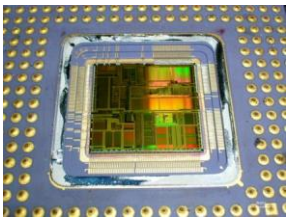
- La puissance d'un processeur se caractérise par le nombre d'instructions qu'il est capable de traiter par seconde.
- On définit alors:
  - Le **CPI** (Cycle Par Instruction)
    - Nombre moyen de cycles d'horloge nécessaire pour l'exécution d'une instruction
  - Le **MIPS** (Millions d'Instructions Par Seconde)
    - Puissance de traitement du processeur

$$MIPS = \frac{F_H}{CPI} \qquad \text{avec } F_H \text{ en MHz}$$

Augmentez la puissance ne passe pas que par l'augmentation de la fréquence d'horloge…



RISC vs CISC



- Il existe deux grandes familles au niveau des jeux d'instructions *(ISA = Instruction Set Architecture)*
  - L'architecture **CISC**
    - Complex Instruction Set Computer
  - L'architecture **RISC**
    - Reduced Instruction Set Computer

RISC vs CISC

Un peu d'histoire [encore]

- Par le passé la conception de machines CISC était la seule envisageable.
- La mémoire travaillait très lentement par rapport au processeur, on pensait donc qu'il était plus intéressant de soumettre au microprocesseur des instructions complexes.
- Ainsi, plutôt que de coder une opération complexe par plusieurs instructions plus petites (qui demanderaient autant d'accès mémoire très lent), il semblait préférable d'ajouter au jeu d'instructions du microprocesseur une instruction complexe qui se chargerait de réaliser cette opération.
- C'est dans les années 80 que la notion de RISC est apparue. Elle consiste à minimiser le nombre d'instructions et à les simplifier!
- Des chercheurs d'IBM, sous la direction de John Cocke se sont convaincus qu'un ensemble réduit d'instructions '' rapides/efficaces'' valait mieux qu'un plus grand ensemble augmenté d'instructions plus lentes et moins efficaces.

RISC vs CISC

Un peu d'histoire [toujours…]

- Des études ont montré que 80% des traitements des langages de haut niveau ne faisaient appel qu'à 20% des instructions d'un processeur.
- L'idée a donc été de réduire le jeu d'instructions et d'améliorer les vitesses de traitement (un cycle d'horloge par instruction).

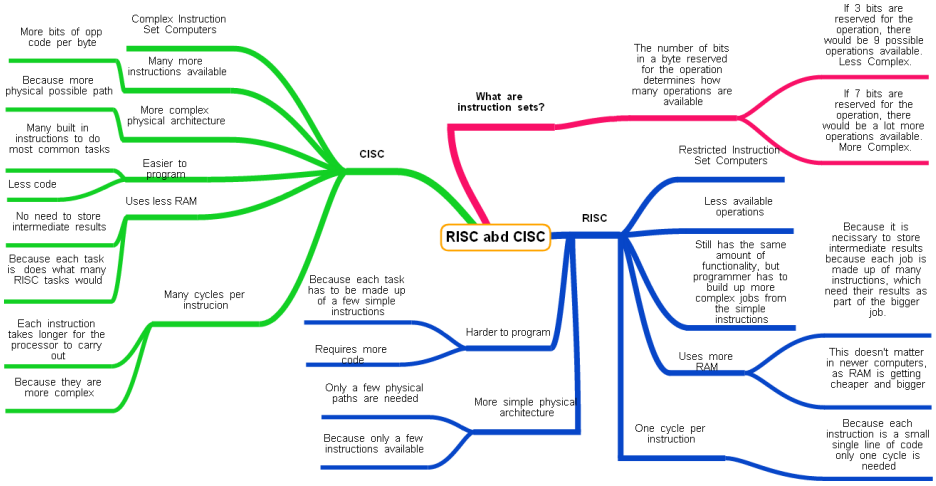
Architecture RISC	Architecture CISC
instructions simples ne prenant qu'un seul cycle instructions au format fixe décodeur simple (câblé) beaucoup de registres seules les instructions LOAD et STORE ont accès à la mémoire peu de modes d'adressage compilateur complexe	instructions complexes prenant plusieurs cycles instructions au format variable décodeur complexe (microcode) peu de registres toutes les instructions sont susceptibles d'accéder à la mémoire beaucoup de modes d'adressage compilateur simple

Et… permet le pipelining…

Source: T. Dumartin

RISC vs CISC

Synthèse



Source: http://a2computing.as93.net/

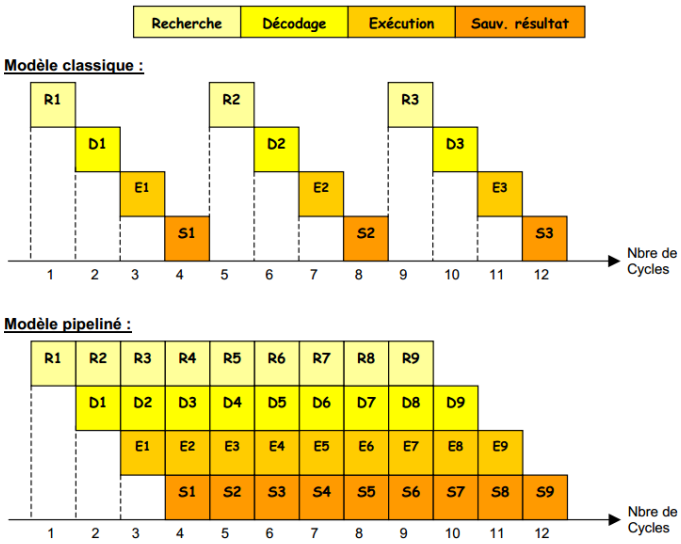
Le pipelining

- L'exécution d'une instruction se décompose en une succession d'étapes
- Chacune de ces étapes fait appel à des sous parties différentes du processeur
- Il est donc possible d'optimiser le traitement des instructions en organisant les traitements sous forme de « pipeline »

Le pipelining

Principe

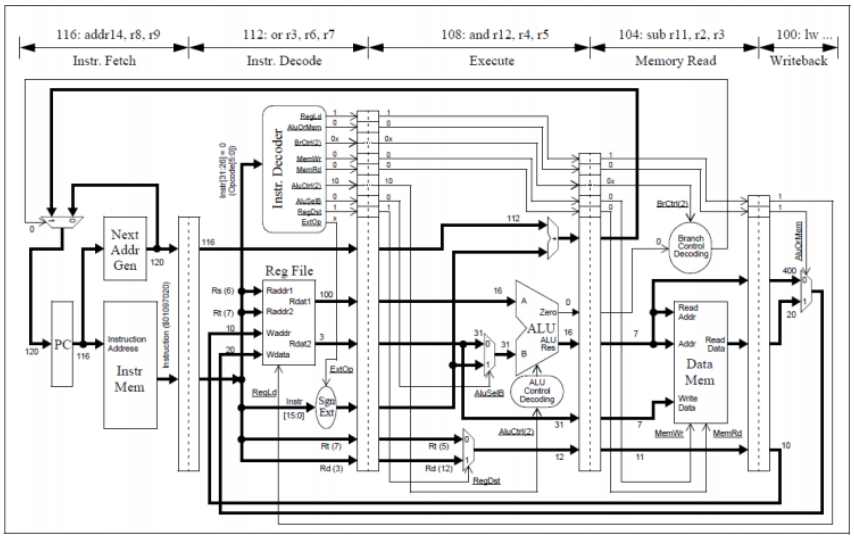
Exemple de l'exécution en 4 phases d'une instruction :



Source: T. Dumartin

Le pipelining

Corrélation avec l'architecture interne du processeur



Source: T. Dumartin

Le pipelining

Retards

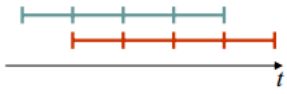
- Le pipeline atteint son plein rendement une fois qu'il est "rempli"
- Un retard peut se produire
  - S'il existe un conflit de ressources (retard ponctuel)
    - accès à la mémoire
    - utilisation des bus
  - En cas de rupture de séquence (vidange du pipeline)
  - branchement non prévu
  - appel de sous-programme
  - interruption

Le pipelining

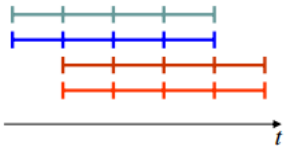
Types de pipelining

Séquentiel :
Pas de pipeline
(ex: Motorola 56000)

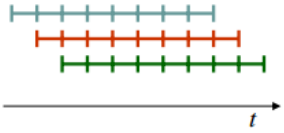
Pipeline simple
(plupart des DSP)



Double pipeline
(ex: Pentium)



Superpipeliné :
Nombre d'étages plus élevé
(ex: TMS320C6000)



Source: Sylvain MONTAGNY

Le pipelining

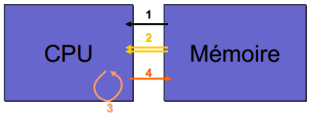
Types d'aléas

- **Aléas de structure** : L'implémentation empêche une certaine combinaison d'opérations (lorsque des ressources matériels sont accédées par plusieurs étages).
- **Aléas de données** : Le résultat d'une opération dépend de celui d'une opération précédente qui n'est pas encore terminée.
- **Aléas de contrôle** : L'exécution d'un saut conditionnel ne permet pas de savoir quelle instruction il faut charger dans le pipeline puisque deux choix sont possibles.

- Il existe des « astuces » pour éviter (limiter) l'impact de ces différents aléas. On citera notamment la **prédiction de branchement** (*branch prediction*) concernant les aléas de contrôle et l'**ordonnancement des instructions** par le compilateur concernant les aléas de données.

Mémoires au pluriel

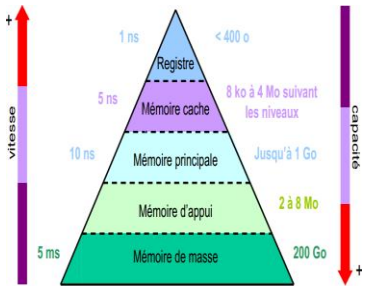
- Plutôt que de parler de **LA** mémoire, il faudrait parler **DES** mémoires…
- En effet…



- (1) Charger une instruction depuis la mémoire
- (2) Charger les opérandes depuis la mémoire
- (3) Effectuer les calculs
- (4) Stocker le résultat en mémoire

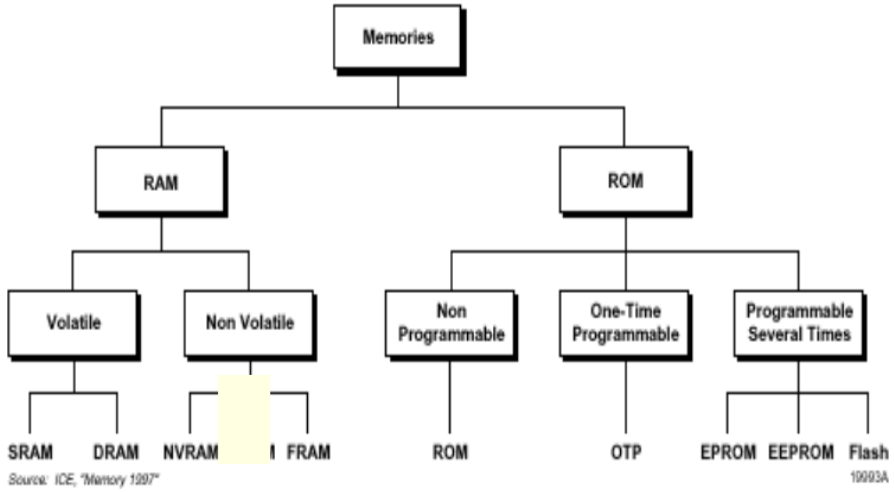
Mémoires au pluriel
Notion de hiérarchie mémoire

- Les **registres** sont les éléments de mémoire les plus rapides. Ils sont situés au niveau du processeur et servent au stockage des opérandes et des résultats intermédiaires.
- La mémoire **cache** est une mémoire rapide de faible capacité destinée à accélérer l'accès à la mémoire centrale en stockant les données les plus utilisées.
- La **mémoire principale** est l'organe principal de rangement des informations. Elle contient les programmes (instructions et données) et est plus lente que les deux mémoires précédentes.
- La **mémoire d'appui** sert de mémoire intermédiaire entre la mémoire centrale et les mémoires de masse. Elle joue le même rôle que la mémoire cache (cache disque)
- La **mémoire de masse** est une mémoire périphérique de grande capacité utilisée pour le stockage permanent ou la sauvegarde des informations. Elle utilise pour cela des supports magnétiques (disque dur) ou optiques (CDROM, DVDROM).



Mémoires au pluriel

Mémoire « vive » vs Mémoire « morte »



Mémoires au pluriel

Encore un peu d'histoire…

- Mémoire « morte »
  - ROM - « Read Only Memory »
  - PROM
  - EPROM
  - EEPROM
  - OTP
- Mémoire « vive »
  - RAM - « Random Access Memory »
  - SRAM
  - DRAM
  - NVRAM
  - Flash NAND - NOR
  - …

Computer memory types

Volatile

RAM
DRAM (e.g., DDR SDRAM) · SRAM

In development
T-RAM · Z-RAM

Historical
Williams–Kilburn tube (1946–47) · Delay line memory (1947) · Selectron tube (1953) · Dekatron

Non-volatile

ROM
Mask ROM · PROM · EPROM · EEPROM

NVRAM
Flash memory · Solid-state storage

Early stage NVRAM
nvSRAM · FeRAM · MRAM · PRAM

Mechanical
Magnetic tape · Hard disk drive · Optical drive

In development
3D XPoint · CBRAM · SONOS · RRAM · Racetrack memory · NRAM · Millipede memory · FJG

Historical
Paper data storage (1725) · Drum memory (1932) · Magnetic-core memory (1949) · Plated wire memory (1957) · Thin-film memory (1962) · Twistor memory (~1968) · Bubble memory (~1970)

V · T · E

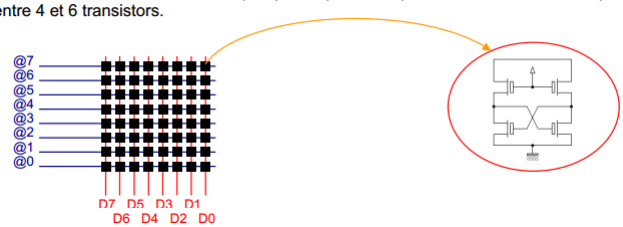
Source: https://en.wikipedia.org/wiki/Computer_memory

Mémoires au pluriel

RAM statique vs RAM dynamique (1/2)

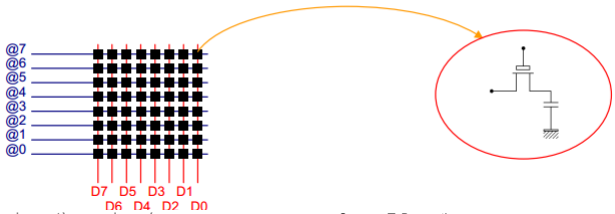
- SRAM

Le bit mémoire d'une RAM statique (SRAM) est composé d'une bascule. Chaque bascule contient entre 4 et 6 transistors.



- DRAM

Dans les RAM dynamiques (DRAM), l'information est mémorisée sous la forme d'une charge électrique stockée dans un condensateur (capacité grille substrat d'un transistor MOS).



Source: T. Dumartin

Mémoires au pluriel

RAM statique vs RAM dynamique (2/2)

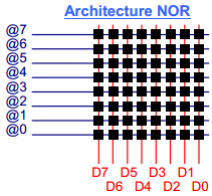
- Les DRAM ont une plus grande densité d'intégration et une consommation réduite
- Des courants de fuites dans les DRAM imposent un rafraîchissement période des informations
- La lecture d'une DRAM est destructrices: une opération de lecture doit donc nécessairement être suivie d'une opération de réécriture

Principalement pour des raisons de coût, la technologie DRAM est utilisée pour la mémoire centrale (grande capacité) tandis que la SRAM est utilisée pour les mémoires dont la vitesse est le critère prioritaire (Cache / Registres)

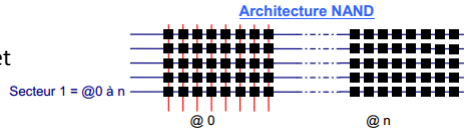
Mémoires au pluriel

Focus sur les types de Flash (1/2)

- NOR
  - Programmation et effacement mot par mot possible
  - Temps d'accès faible (octet)
  - Temps d'accès long (paquets)
  - Coût élevé



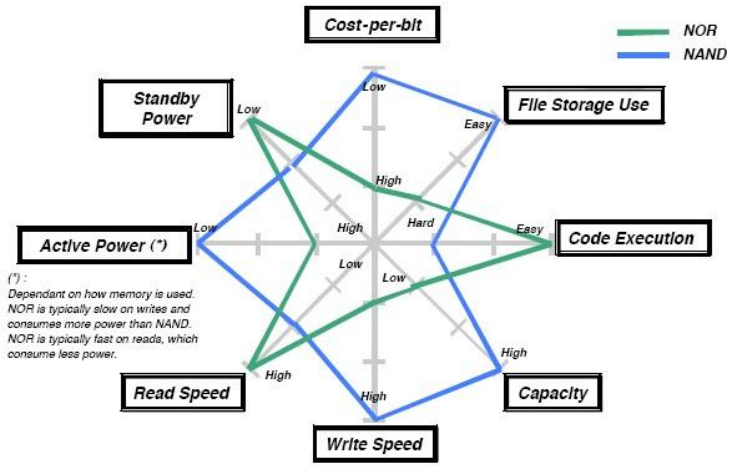
- NAND
  - Forte densité d'intégration
  - Coût réduit
  - Rapidité de lecture / écriture par paquet
  - Consommation réduite
  - Lecture / écriture par octet impossible
  - Interface E/S indirecte



Mémoires au pluriel

Focus sur les types de Flash (2/2)

Fig. 1 Comparison of NOR and NAND Flash



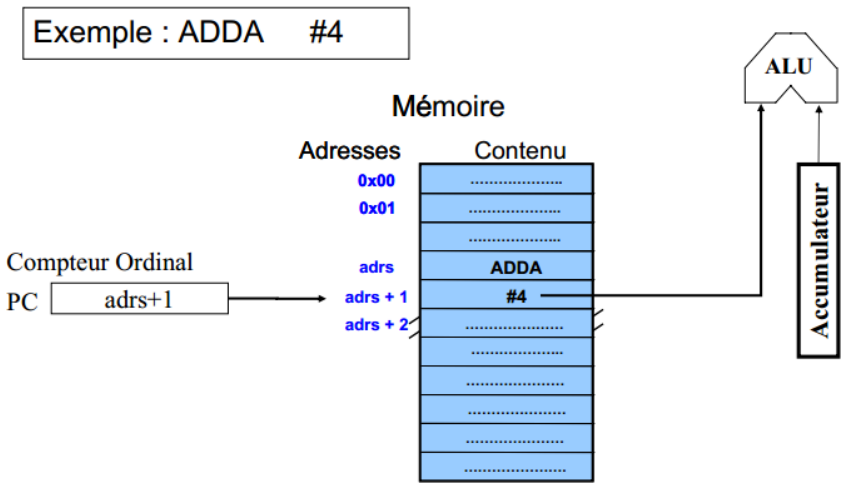
 Source: http://embeddeddomain.blogspot.fr/2011/04/nor-vs-nand-flash.html

Mémoires au pluriel
Modes d'adressage (1/4)

- Un mode d'adressage définit la manière dont le microprocesseur va accéder à l'opérande. Les différents modes d'adressage dépendent des micros mais on retrouve en général :
  - l'adressage de registre où l'on traite la données contenue dans un registre
  - l'adressage **immédiat** où l'on définit immédiatement la valeur de la donnée
  - l'adressage **direct** où l'on traite une données en mémoire
  - l'adressage **indirect** où l'on traite l'adresse d'une donnée en mémoire

 Source: http://embeddeddomain.blogspot.fr/2011/04/nor-vs-nand-flash.html

Mémoires au pluriel
Modes d'adressage (2/4)- Adressage immédiat

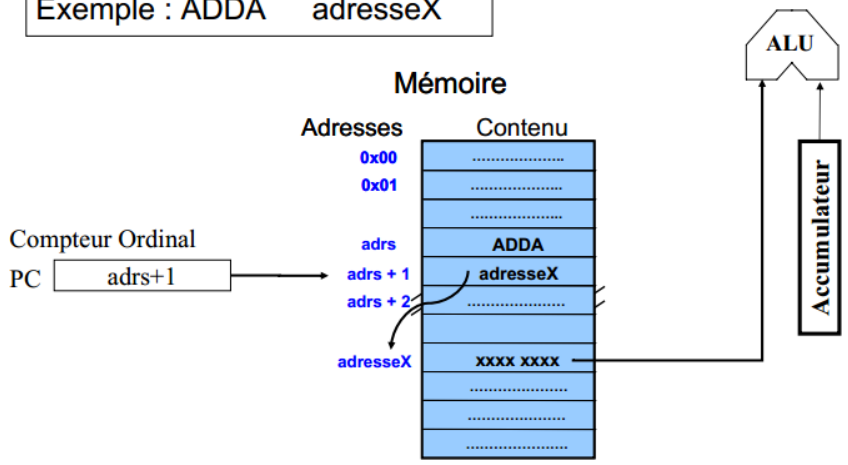


 Source: Sylvain MONTAGNY

Mémoires au pluriel

Modes d'adressage (3/4)- Adressage direct

Exemple : ADDA adresseX

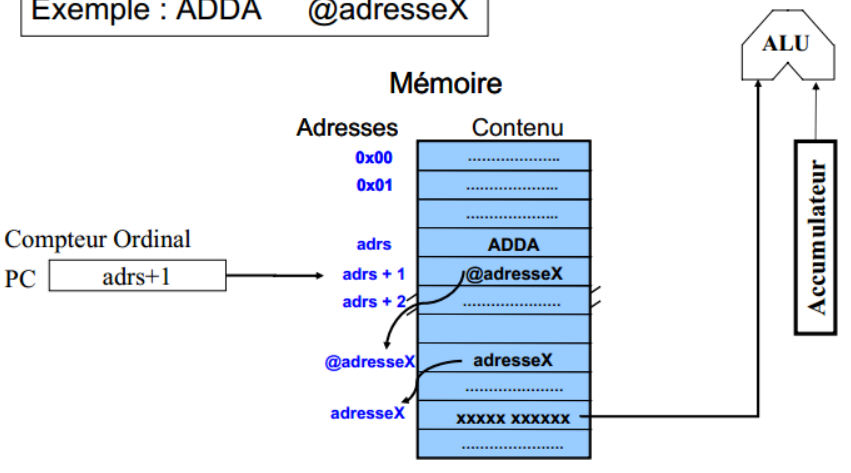


Source: Sylvain MONTAGNY

Mémoires au pluriel

Modes d'adressage (4/4)- Adressage indirect

Exemple : ADDA @adresseX

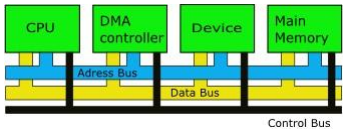


Source: Sylvain MONTAGNY

Mémoires au pluriel

Le principe du DMA - Direct Memory Access (1/2)

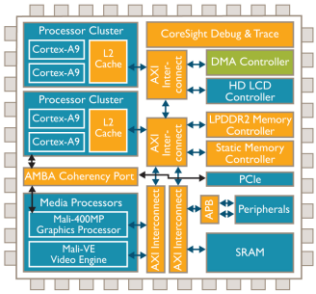
- Le système doit récupérer des données en provenance de périphériques externes.
- Plusieurs méthodes sont possibles :
  - Une méthode par scrutation (**polling**) permet d'interroger régulièrement les périphériques afin de savoir si une nouvelle donnée est présente.
  - Une méthode par **interruption** permet au périphérique lui-même de faire signe au processeur de sa présence.
  - Une méthode par **Accès Direct à la Mémoire** (DMA) permet de gérer le transfert de façon autonome.

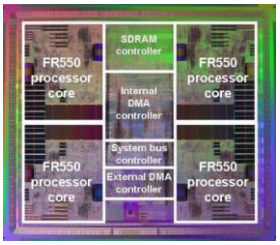


Mémoires au pluriel

Le principe du DMA - Direct Memory Access (2/2)

- L'accès direct à la mémoire ou DMA est un procédé où des données circulant de ou vers un périphérique (port de communication, disque dur) sont transférées directement par un contrôleur adapté vers la mémoire centrale de la machine, sans intervention du microprocesseur.
- Le micro interviendra seulement pour initier et conclure le transfert. La conclusion du transfert ou la disponibilité du périphérique peuvent être signalés par interruption.
- Le transfert DMA nécessite un circuit appelé contrôleur de DMA.





(Source : Wikipédia)

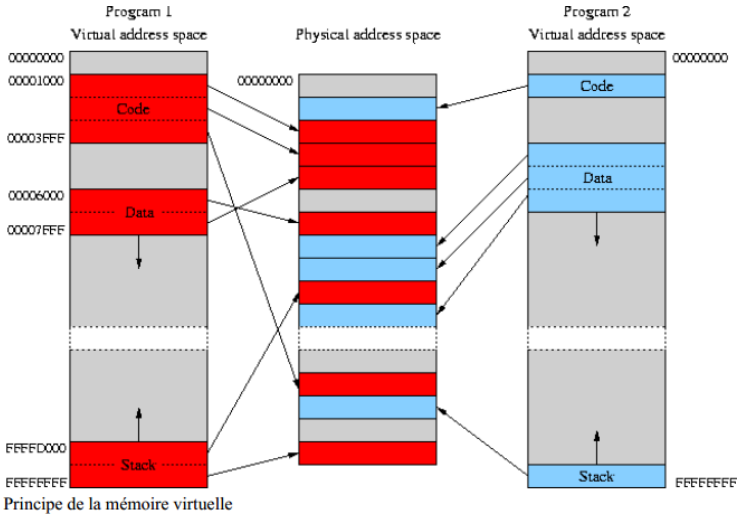
Mémoires au pluriel

Mémoire virtuelle (1/3)

- Lors de la compilation d'un programme écrit dans un langage de haut niveau, les références à des **variables locales ou globales** sont traduites en emplacements mémoire alloués à ces variables.
- Variables globales dans la **zone des données**
- Variables locales dans la **pile**
- Dans un processeur sans MMU (**Memory Management Unit**), les adresses manipulées sont les adresses réelles des données en mémoire. On parle d'**adresses virtuelles** et d'**adresses physiques**.
- Dans un processeur avec MMU, chaque programme dispose d'un **espace d'adressage virtuel**.
- La correspondance entre adresses virtuelles et adresses physiques est faite conjointement par le gestionnaire de MMU et le système d'exploitation.

Mémoires au pluriel

Mémoire virtuelle (2/3)

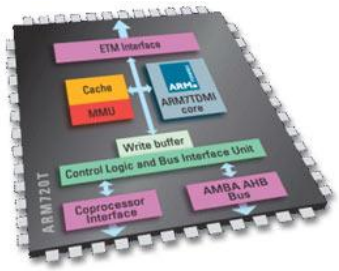


Principe de la mémoire virtuelle

Source: Olivier Carton

Mémoires au pluriel

Mémoire virtuelle (3/3)



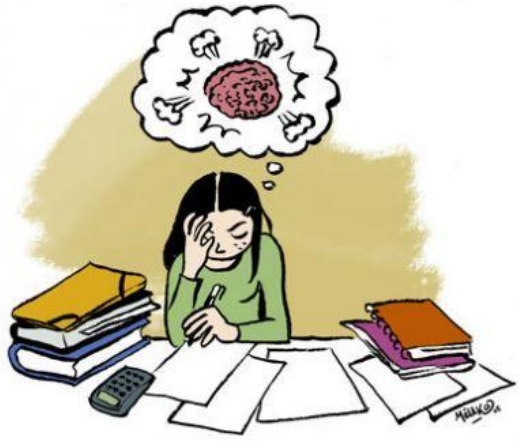
- Fonctions d'une MMU
  - Traduction d'adresses virtuelles en adresses physiques
  - Contrôle de tampon
  - Arbitrage du bus
  - Protection de la mémoire (MPU = Memory Protection Unit)
  - Commutation de banque (principalement dans les systèmes à espace d'adressage physique limité)
  - Allocation d'un espace d'adressage virtuel contigu même en cas de fragmentation de la mémoire physique libre.

Mémoires au pluriel

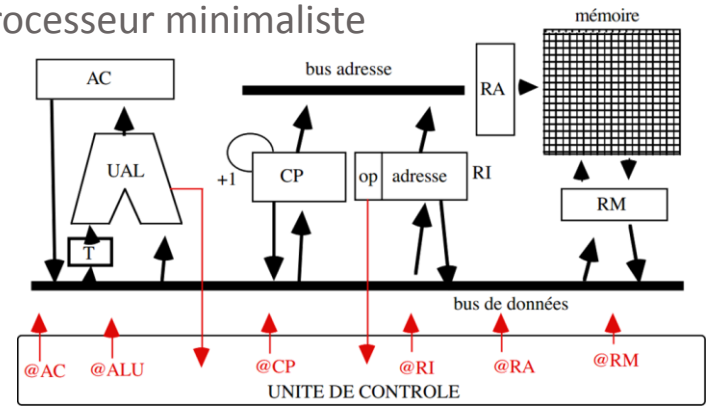
Une zone mémoire particulière: la pile

- La pile (= stack), appelée également **pile d'exécution** est un espace mémoire particulier
- Cette zone mémoire, gérée en mode **LIFO** (Last In First Out), permet de stocker:
  - les paramètres d'appel des fonctions
  - la valeur des registres lors des changements de contexte
  - les valeurs des variables locales
  - la valeur de retour des fonctions
  - les adresses de retour des fonctions
- La durée de vie de ce qui est stocké dans la pile n'excède pas la **durée d'exécution d'une fonction**
- Si l'espace mémoire dédiée à la pile devient insuffisant, il y a alors un **dépassement de pile** ou **débordement de pile** (**stack overflow**)

Exercice



Un processeur minimaliste



Source: CNAM

- Rappeler le rôle des différents organes
- Quelle est la taille de l'espace mémoire adressable ?
- Quels sont les nombres manipulables par l'UAL ?
- Décrire le séquencement de l'instruction ADD 155
- Décrire le séquencement de l'instruction ADDi 155

Rupture dans l'exécution des instructions
Les interruptions (1/11)

- Un système informatique n'est utile que s'il communique avec l'extérieur. L'objectif est de pouvoir prendre connaissance que le périphérique sollicite le processeur. Cette sollicitation arrive de façon totalement asynchrone.

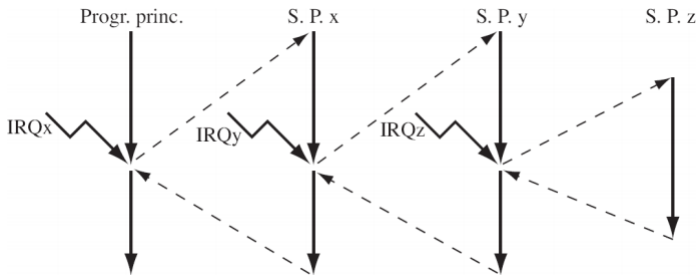
- Deux modes sont possibles :
  - Une méthode par **scrutation** (polling) permet d'interroger régulièrement les périphériques afin de savoir si une nouvelle donnée est présente.
  - Une méthode par **interruption** permet au périphérique lui même de faire signe au processeur de sa présence.

Rupture dans l'exécution des instructions
Les interruptions (2/11)

- Scrutation vs Interruption
  - Scrutation (polling)
    - Coûteux en temps (multiplier par le nombre de périphérique à interroger)
    - Implémentation : Appel classique à une fonction dans le programme
  - Interruption
    - Demande à l'initiative du périphérique
    - Prise en compte rapide de l'évènement
    - Implémentation : Interruption asynchrone d'un programme puis retour au même endroit à la fin du traitement

Rupture dans l'exécution des instructions
Les interruptions (3/11)

- Une **interruption** est un arrêt temporaire de l'exécution normale d'un programme informatique par le microprocesseur afin d'exécuter un autre programme (appelé routine d'interruption / sous-programme d'interruption).
- Exemple d'interruptions imbriquées (**nested** interrupts):



Source: Sylvain MONTAGNY

Rupture dans l'exécution des instructions
Les interruptions (4/11)

- Types d'interruption
  - Interruption masquable
    - Un masque d'interruption (IMR = Interrupt Mask Controller) est un mot binaire de configuration du micro qui permet de choisir (**masquer** / **démasquer**) quels modules pourront interrompre le processeur parmi les interruptions disponibles
  - Interruption non masquable (NMI = Non Maskable Interrupt)
    - Elles s'exécutent quoi qu'il arrive, souvent avec une priorité élevée (ex: Reset)

Rupture dans l'exécution des instructions
Les interruptions (5/11)

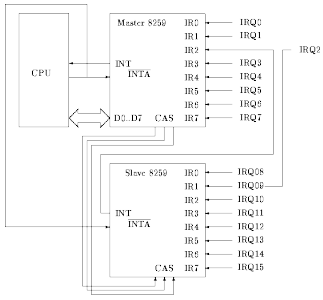
- Configuration
  - Un système peut accepter plusieurs sources d'interruption. Chacune est configurable par le registre d'interruption (IRR = Interrupt Request Register)
  - Méthode:
    - Sélectionner les interruptions intéressantes
    - Valider les interruptions de façon globale
    - Ecrire les sous programmes d'interruption
    - Définir les priorités entre interruptions

Rupture dans l'exécution des instructions
Les interruptions (6/11)

- Que se passe t'il dans les routines d'interruption (**ISR = Interrupt Service Routine**)?
  - Sauvegarde du contexte dans la pile (sauvegarde de la valeur des registres)
  - Définition de la source de l'interruption
  - Réinitialisation des flags d'interruption
  - Code relatif à la prise en compte de l'interruption
  - Restitution du contexte

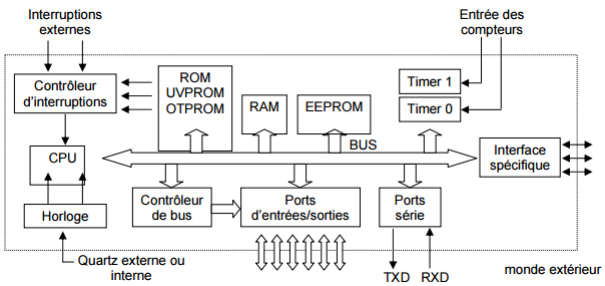
Rupture dans l'exécution des instructions
Les interruptions (7/11)

- Et au niveau du matériel?
  - Présence d'un **gestionnaire d'interrruption (PIC = Programmable Interrupt Controller)**
    - Ex: le PIC originel des IBM / PC
      - 2 x Intel 8259



Rupture dans l'exécution des instructions
Les interruptions (8/11)

- Et au niveau matériel?
  - Présence d'un **gestionnaire d'interrruption (PIC = Programmable Interrupt Controller)**
    - Autre exemple: architecture interne d'un microcontrôleur

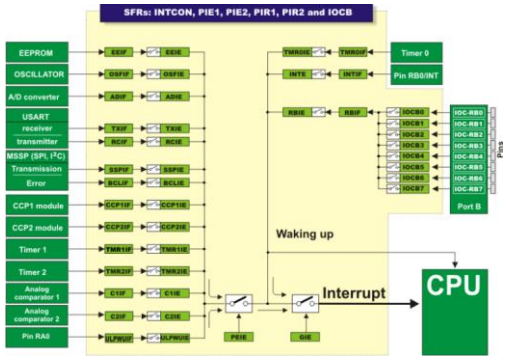


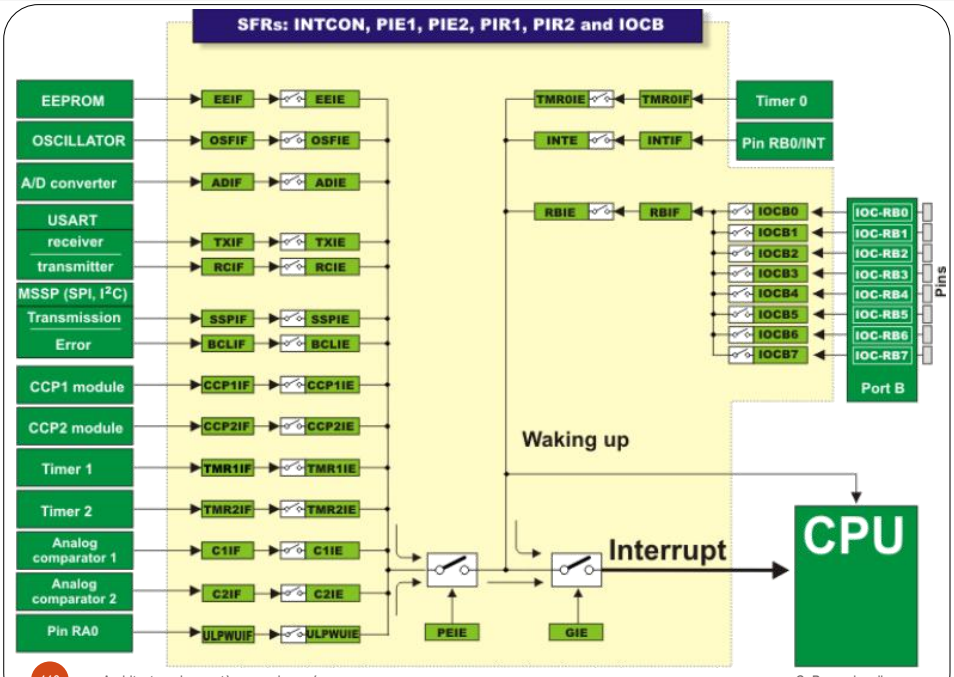
Source: Jérôme VICENTE

Rupture dans l'exécution des instructions

Les interruptions (9/11)

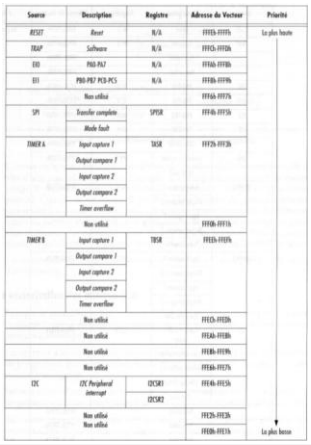
- Et au niveau matériel?
  - Présence d'un **gestionnaire d'interrruption (PIC = Programmable Interrupt Controller)**
    - Autre exemple: gestion des interruption PIC Microchip



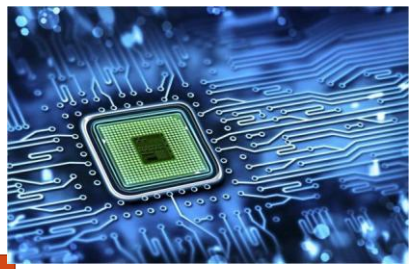


Rupture dans l'exécution des instructions
Les interruptions (11/11)

- Vecteurs d'interruption
  - Appelés aussi « table d'interruption »
  - Contient les adresses des routines d'interruption (ISR) correspondantes à chaque source d'interruption
  - Exemple:
    table d'interruption du ST7 →



Plan

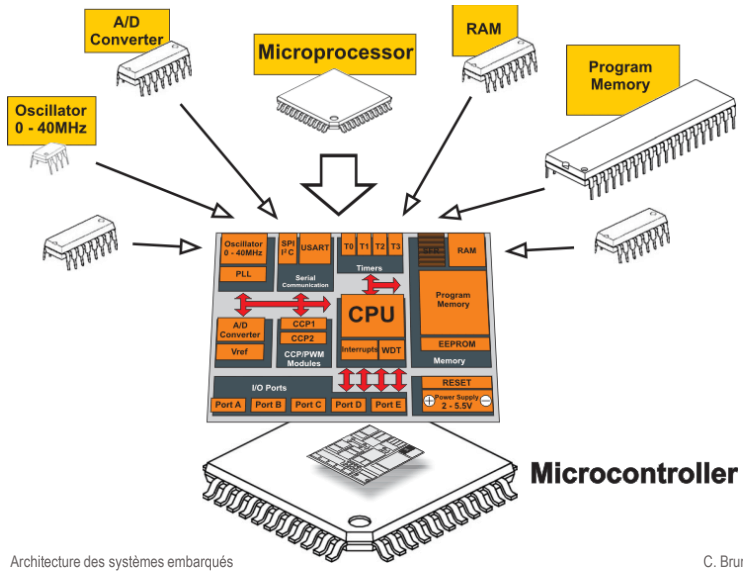


- Introduction
- Un peu d'histoire…
- Rappels théoriques
- **Diversité des « puces électroniques »**
- Familles de micros
- Mise en œuvre d'un STM32 à base d'ARM Cortex M4
- Conception des systèmes embarqués
- Du code au monde physique (par l'étude de quelques cartes – systèmes électroniques)
- Cas concrets: analyse de différents systèmes embarqués

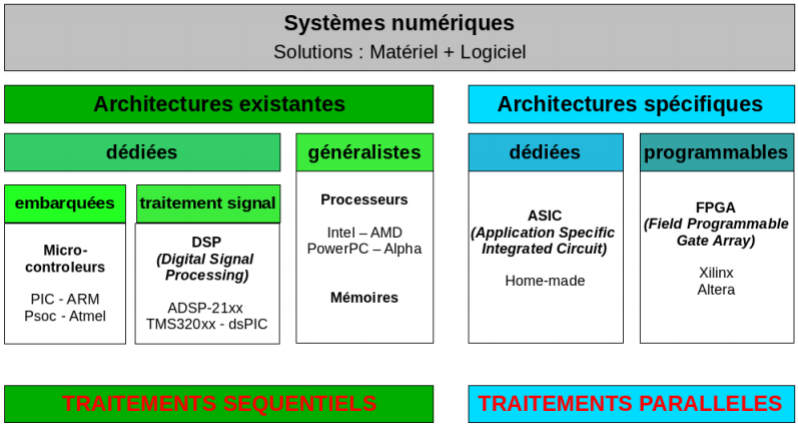
Diversité des « puces électroniques »

Logique programmable
Processeurs généraux
Processeurs spécialisés
Processeurs dédiés / Microcontrôleurs
SoC (System On Chip)
Mono - Multi - Many Cœur(s)

Il n'y a pas que Microprocesseur vs Microcontrôleur



Il n'y a pas que Microprocesseur vs Microcontrôleur



Source: Julien Villemejane

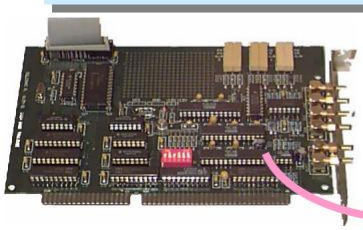
Logique programmable

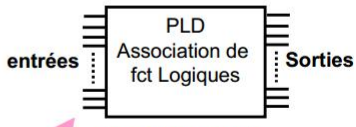
Logique programmable

Circuits logiques programmables

Circuits logiques + ou - complexes remplaçant l'association de plusieurs boîtes logiques

- Simplification de câblage
- Encombrement réduit
- Diminution des coûts
- Facilité d'utilisation
- Diminution du risque des pannes





1 seul circuit intégré

 Source: Fabrice Caignet

Logique programmable

ASIC vs PLD (FPGA)

ASIC
Application Specific Integrated Circuits

- Choix du fondeur
- Conception du circuit (full-custom - bibliothèques)
- Très grand niveau d'intégration
- Fabrication à très grand nombre d'exemplaire

- Un circuit dédié à une application
- Choix de la technologie
  ⇨ Maximum de performances

PLD
Programmable Logic Device

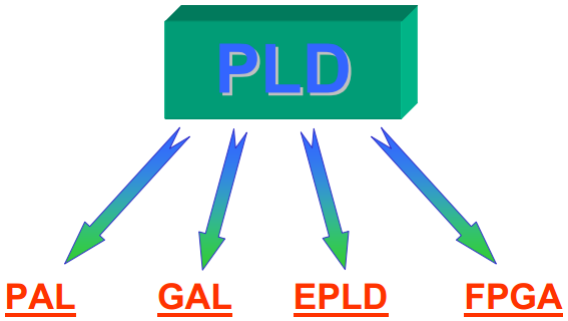
- Choix du circuit
- Programmation du circuit (logiciel + interface circuit)
- Intégration limitée
- Implémentation nombre d'exemplaires limité

- Un circuit programmé pour une application
- Technologie figée mais :
  ⇨ Facilité de programmation

 Source: Fabrice Caignet

Logique programmable

Quatre familles de PLD

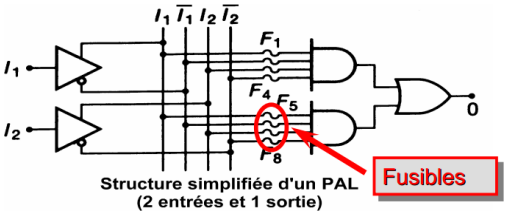


- PAL = Programmable Array Logic
- GAL = Generic Array Logic (= PAL effaçable électriquement)
- EPLD = Erasable Programmable Logic Device
- FPGA = Field Programmable Gate Array

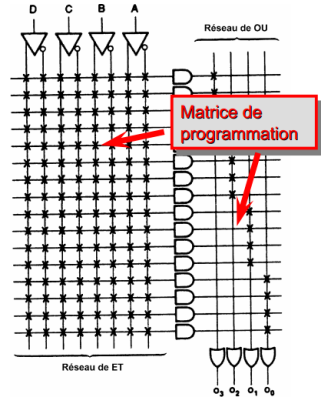
Pour info: CPLD = Complex Programmable Logic Device

Logique programmable

Pour comprendre un peu mieux…

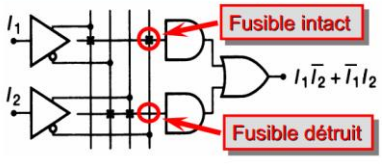


Structure simplifiée d'un PAL
(2 entrées et 1 sortie)



Représentation :

- les fusibles intactes sont représentés par une connexion
- les fusibles détruits sont représentés par une absence de connexion



Exemple d'un "OU EXCLUSIF"

Logique programmable

Les FPGA - Introduction

- Les FPGA sont des circuits programmables assimilables à de nombreux (C)PLD mis ensembles sur une puce

- Deux fabricants principaux:
  - ALTERA
  - XILINX
- Langages de programmation pour les FPGA - HDL (Hardware Description Language)
  - VHDL = Very High Speed integrated circuit HDL, proche de l'Ada
  - Verilog, proche du C

Logique programmable

Les FPGA - Exemple de fonctions implémentables

- Contrôleur de communication
  - USB (host / device), Ethernet, SPI, I2C, UART, etc..
- Fonctions arithmétiques + ou - complexes
  - ALU / FPU
- Fonctions de sécurité
  - AES encryption / decryption, MD5, SHA, etc…
- Fonctions de traitement du signal
  - FFT, Filtres numériques, PID, etc…
- Traitements multimédia
  - Compression, décompression vidéo/audio, contrôleur HDMI, etc…
- Etc…

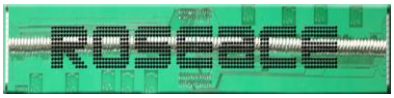
OpenCores
www.opencores.org

fpga4fun.com - where FPGAs are fun.

Logique programmable

Les FPGA - Just for fun (…)

- Projet de 3 étudiants de Paritech (2012)
- 112 leds RGB
- 1 FPGA (ALTERA Cyclone III)
- Et voilà le résultat:



http://roseace.telecom-paristech.fr/

Processeurs généraux

- Fonctionnalités multiples
- Parfaitement surdimensionnés pour les systèmes embarqués
- Exemples:
  - Famille Intel x86
  - Famille AMD compatible x86
  - Famille Motorola/IBM PowerPC
  - … (tout processeur équipant un ordinateur généraliste)

Processeurs spécialisés

- Processeurs de traitement du signal (DSP)
  - Exemple: famille TMS320 de Texas Instrument

- Processeurs graphiques (GPU)
  - Structure hautement parallèle
  - Ex: 192 cœurs CUDA dans un Nvidia Tegra K1

Processeurs dédiés / Microcontrôleurs

- Des milliers de références!
- Quelques grandes familles
  (certaines seront présentées dans le chapitre suivant)

SoC (System On Chip)

Définitions

- Microcontrôleur:

"A microcontroller (sometimes abbreviated µC, uC or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals."

- SoC:

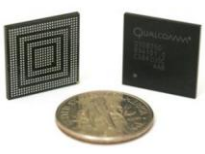
"A system on a chip or system on chip (SoC or SOC) is an integrated circuit (IC) that integrates all components of a computer or other electronic system into a single chip."

SoC (System On Chip)

Définitions

- Finalement, pas de définition complètement figée
- Distinction plus marketing que technique entre MCU et SoC
- Si on avait à faire une distinction:
  - MCU:
    Historique remontant aux années 70 pour désigner une puce intégrant à la fois un MPU associé à de la mémoire (programme + data) et quelques entrées / sorties.

  - SoC:
    Historique plus récent. Vient du monde des ASIC et désigne des puces dans lequel on retrouve tout ce qui est nécessaire pour construire un « système » complet. De façon générale, un SoC va être plus spécifique à une application qu'un MCU. Un SoC peut contenir à la fois des parties digitale, analogiques ou mix.



Mono - Multi - Many Cœur(s)

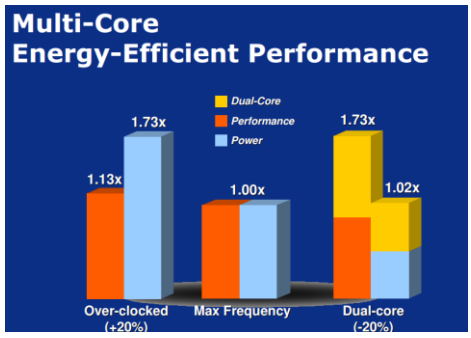
- Recourir à plusieurs cœurs de micro est venu d'un besoin de toujours plus de puissance de calculs…
- … mais pas seulement…

- Au final, on pourrait identifier (au moins) cinq raisons:
  - Besoin de puissance
  - Besoin de parallélisme
  - Besoin de sûreté de fonctionnement
  - Besoin d'optimiser l'énergie
  - Besoin de répondre à des besoins hétéroclites

Mono - Multi - Many Cœur(s)

Besoin de puissance de calcul

- Multiplier la fréquence de fonctionnement a atteint ses limites physiques
- En implantant plusieurs CPU dans un même composant, on multiplie la puissance de calcul, à fréquence égale.



Source: Baskaran Ganesan

Mono - Multi - Many Cœur(s)

Besoin de parallélisme

- Dans la plupart des applications, il s'agit de traiter différentes tâches en parallèle.
  - Par exemple, gestion d'une interface homme machine, tout en traitant l'émission / réception de données sur un bus/réseau de communication
- De même, certaines applications telles que le traitement ou la génération d'images, ainsi que les traitements vidéos nécessitent de réaliser des traitements en parallèle.
- Exemples:
  - les processeurs graphiques many cœurs de Nvidia
  - le processeur many cœurs de Kalray

Mono - Multi - Many Cœur(s)

Besoin de sûreté de fonctionnement

- Dans certaines applications critiques, le besoin en sûreté de fonctionnement est tel qu'il est nécessaire de mettre en œuvre des principes de redondance.
- C'est le cas, par exemple, des processeurs multi cœurs fonctionnant en « Lock Step ».
- La redondance ainsi créée peut permettre de détecter des erreurs et dans certains cas, les corriger.
- Exemple: famille SPC5 de ST

ASIL-D compliant SPC56 L-line and SPC57 S-line Devices

The new SPC56 L-line (SPC56EL60 and SPC56EL70) and SPC57 S-line (SPC570S40 and SPC570S50) devices comply with the most stringent automotive safety standards (ISO 26262). Featuring an increased non-volatile memory size, these lines are designed so that customers using existing parts can migrate their platforms to these new devices very simply. These devices cover a wide range of automotive applications that need to respect the automotive safety integrity levels (ASIL) that are now required, up to and including the most stringent ASIL-D level. ASIL-D classification is now commonplace in critical systems such as anti-lock braking, electric power steering, active suspension and advanced driver assistance systems (ADAS). The dual-core architecture of the L- and S-line devices reduces duplication of components at a system level, lowering overall system costs. Their architecture also provides unique flexibility by allowing the user to select lockstep or dual parallel processing (independent core operation) modes, enabling support of multiple safety architectures that the user can configure to achieve the required balance between safety and performance levels.

These SPC56EL60 and SPC56EL70 devices are designed to cover a wide range of automotive applications that need to respect the automotive safety integrity levels (ASIL) that are now required, up to and including the most stringent ASIL-D level. ASIL-D is now a commonplace in critical systems such as anti-lock braking, electric power steering, active suspension and advanced driver assistance systems (ADAS).

The SPC56EL60L3, SPC56EL60L5 and SPC56EL70L5 devices combine:

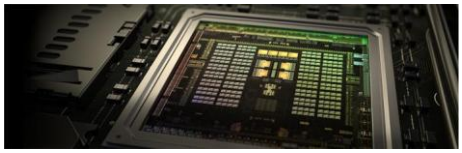
- Two high-performance e200z4d cores
- 1 MB (SPC56EL60) or 2 MB (SPC56EL70) of Flash memory
- 192 KB RAM Flash memory
- Three CAN interfaces
- An optimized peripheral set for safety and motor-control applications, supporting up to two brushless 3-phase motors

The dual-core architecture reduces duplication of components at a system level, lowering overall system costs. The architecture also provides unique flexibility by allowing the user to select lockstep or dual parallel processing (independent core operation) modes, enabling support of multiple safety architectures that the user can configure to achieve the required balance between safety and performance level.

Mono - Multi - Many Cœur(s)

Besoin d'optimiser l'énergie

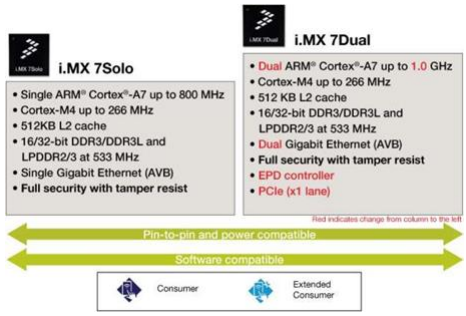
- Comment concilier puissance de calcul et gestion d'énergie?
- La plupart des micros dédiés aux systèmes embarqués sur batterie disposent de mécanismes permettant d'optimiser la consommation en énergie
- Par exemple, modulation de la fréquence de fonctionnement selon les besoins en puissance à un instant t
- Certains micros récents vont jusqu'à embarquer plusieurs types de cœurs.
- Exemple: le tegra X1 de Nvidia (quad core A57 + quad core A53)



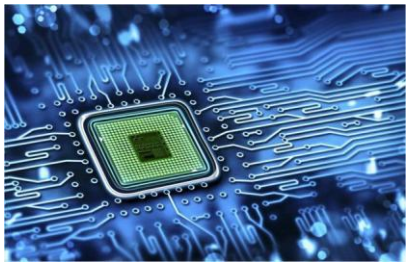
Mono - Multi - Many Cœur(s)

Besoin de répondre à des besoins hétéroclites

- Comment gérer des besoins hétéroclites au sein d'un même système?
- En intégrant différents types de cœurs, chacun répondant à des besoins spécifiques.
- Exemple: l'iMX7 de freescale



Plan



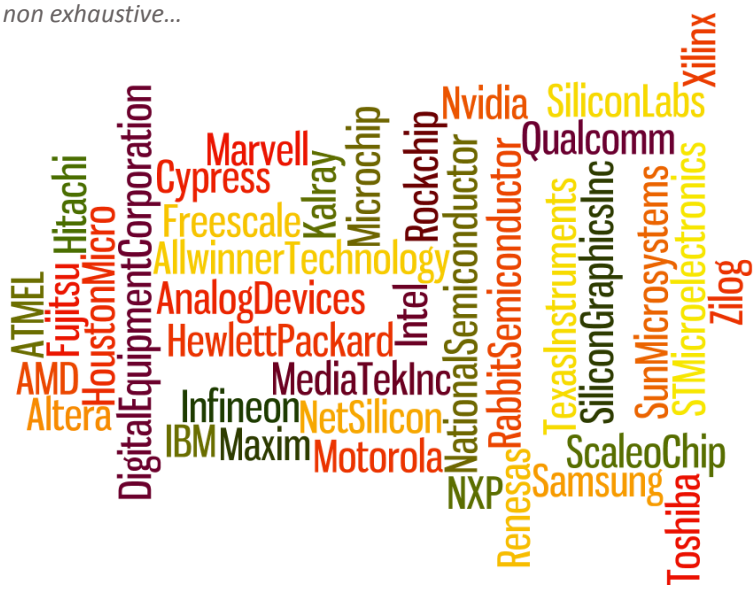
- Introduction
- Un peu d'histoire…
- Rappels théoriques
- Diversité des « puces électroniques »
- **Familles de micros**
- Mise en œuvre d'un STM32 à base d'ARM Cortex M4
- Conception des systèmes embarqués
- Du code au monde physique (par l'étude de quelques cartes – systèmes électroniques)
- Cas concrets: analyse de différents systèmes embarqués

Avant de commencer… un petit brainstorm…

- Quel(s) fondeur(s) connaissez-vous?
- Quelle(s) famille(s) de micros connaissez-vous?

Fondeurs

→liste non exhaustive…



Familles de micros

→liste non exhaustive…

Familles de micros

La famille des 8051

La grande famille des PIC

La **très** grande famille des ARMs

Quelques x86 (car pas très « embedded »…)

Quelques mots sur les MIPS, les TriCore, les pSoC, les TMS320, etc…

La famille des 8051

Une « veille » et grande famille toujours d'actualité (1/2)

"Despite strong media coverage of the rapid expansion of the ARM ecosystem, the largest ecosystem in MCUs still exists around the mature and tiny 8051 MCU architecture."

IHS*, 2013

**IHS Inc. (IHS) is a company based in Douglas County, Colorado, United States. IHS provides information and analysis to support the decision-making process of businesses and governments in industries, such as aerospace, defense and security; automotive; chemical; energy; maritime and trade; and technology.*

Source: wikipedia

La famille des 8051

Une « veille » et grande famille toujours d'actualité (2/2)

Derivate vendors [edit]

Current vendors of MCS-51 compatible processors include more than 20 independent manufacturers including Atmel, Infineon Technologies (formerly Siemens AG), Maxim Integrated Products (via its Dallas Semiconductor subsidiary), NXP (formerly Philips Semiconductor), Microchip Technology, Nuvoton (formerly Winbond), ST Microelectronics, Silicon Laboratories (formerly Cygnal), Texas Instruments, Ramtron International, Silicon Storage Technology, Cypress Semiconductor and Analog Devices.[13]

ICs or IPs compatible with the MCS-51 have been developed by:

- Acer Labs
- Actel
- Aeroflex UTMC
- Altium
- Analog Devices
- ASIX
- Atmel
- AustriaMicroSystems
- AXSEM
- California Eastern Laboratories (CEL)
- Cast
- CML Microcircuits
- CORERIVER
- Cybernetic Micro Systems
- CybraTech
- Cypress Semiconductor
- Daewoo
- Dallas Semiconductor
- Digital Core Design
- Dolphin Integration
- Domosys
- easyplug
- EnOcean
- Evatronix
- Fairchild Semiconductor
- Genesis Microchip
- Genesys Logic
- Goal Semiconductor
- Handshake Solutions
- Honeywell
- Hynix Semiconductor
- Infineon
- InnovASIC
- Intel
- ISSI
- Lapis Semiconductor
- Maxim (Dallas Semiconductor)
- Megawin
- Mentor Graphics
- Micronas
- Microsemi
- MXIC (Macronix)
- Myson Technology
- Nordic Semiconductor
- Nuvoton (Winbond)
- NXP (founded by Philips)
- OKI
- Oregano Systems
- Palmchip
- Prolific
- Radio Pulse
- Ramtron
- RDC Semiconductor
- Sanyo
- Sharp
- Sigma Designs
- Silicon Laboratories (Cygnal)
- Siliconians
- SMSC
- SST
- STMicroelectronics
- SyncMOS
- Synopsys
- Syntek Semiconductor
- Tekmos
- Teridian Semiconductor
- Texas Instruments
- Tezzaron Semiconductor
- Triscend
- Vitesse
- Yitran
- Zensys
- Zilog
- Zylogic Semiconductor

Source: wikipedia

La famille des 8051

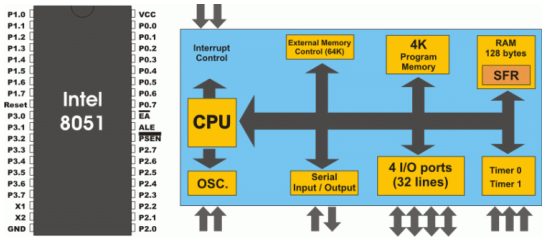
Une longue histoire… raconté brièvement

- In 1981, Intel Corporation introduced an 8-bit microcontroller called the 8051 (first device in the MCS-51® family of 8-bit microcontrollers).
- This microcontroller had 128 bytes of RAM, 4K bytes of on-chip ROM, two timers, one serial port, and four ports (each 8-bits wide) all on a single chip.
- The 8051 is an 8-bit processor, meaning that the CPU can work on only 8 bits of data at a time.
- Data larger than 8 bits has to be broken into 8-bit pieces to be processed by the CPU.

Source: http://www.jefflamoon.com/

La famille des 8051

Une longue histoire… raconté brièvement



- The 8051 has a total of four I/O ports, each 8 bits wide.
- Although the 8051 can have a maximum of 64K bytes of on-chip ROM, many manufacturers have put only 4K bytes on the chip.
- The 8051 became widely popular after Intel allowed other manufacturers to make and market any flavors of the 8051 they please with the condition that they remain code-compatible with the 8051.
- This has led to many versions of the 8051 with different speeds and amounts of on-chip ROM marketed by more than half a dozen manufacturers.

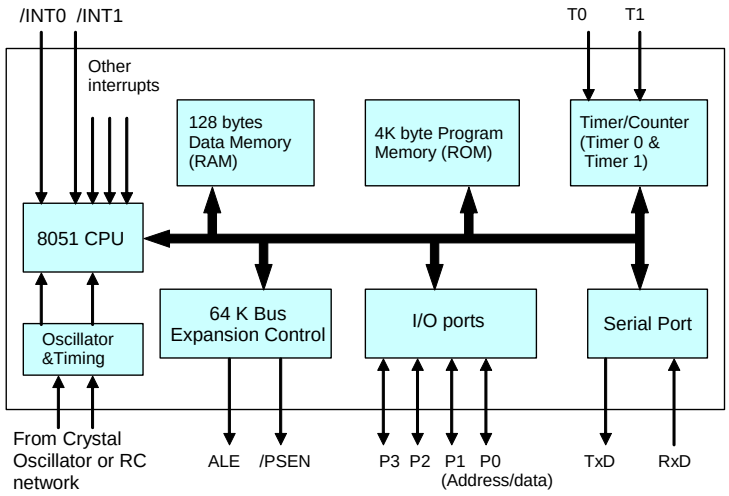
La famille des 8051

Membres de la famille (au départ)

- 8052
  - The 8052 is another member of the 8051 family.
  - The 8052 has all the standard features of the 8051 as well as an extra 128 bytes of RAM and an extra timer.
  - 8052 has 256 bytes of RAM and 3 timers.
  - It has 8K bytes of on-chip program ROM instead of 4K bytes.
- 8031
  - This chip is often referred to as a ROM-less 8051 since it has 0K bytes of on-chip ROM.
  - To use this chip you must add external ROM to it.
  - This external ROM must contain the program that the 8031 will fetch and execute.
  - The ROM containing the program attached to the 8031 can be as large as 64K bytes.
  - In the process of adding external ROM to the 8031, you lose two ports.
  - To solve this problem, you can add external I/O to the 8031, such as the 8255 chip.

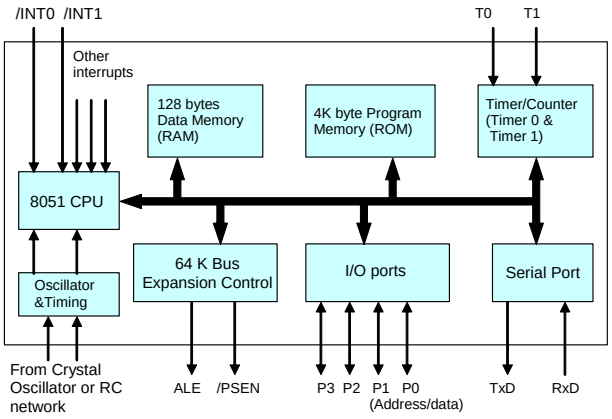
La famille des 8051

Architecture générale



La famille des 8051

Harvard ou Von Neumann ??



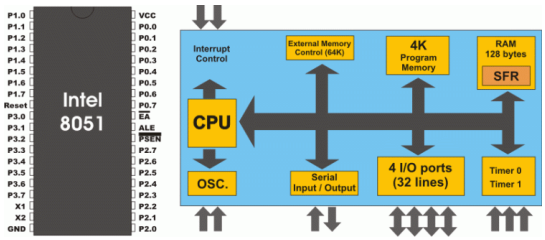
Harvard and von Neumann Architectures

- Harvard Architecture - a type of computer architecture where the instructions (program code) and data are stored in separate memory spaces
  - Example: **Intel 8051 architecture**
- von Neumann Architecture - another type of computer architecture where the instructions and data are stored in the same memory space
  - Example: Intel x86 architecture (Intel Pentium, AMD Athlon, etc.)

Source: Silicon Labs

La famille des 8051

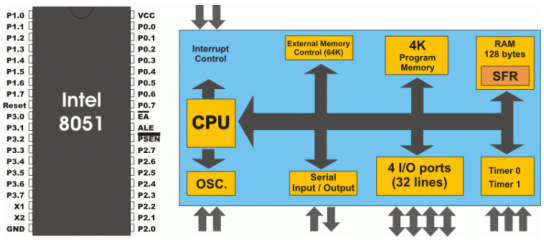
Pins description (1/4)



- VCC & GND : Power supply
- Reset
  - This is an input pin
  - A logical “1” asks a reset to the CPU

La famille des 8051

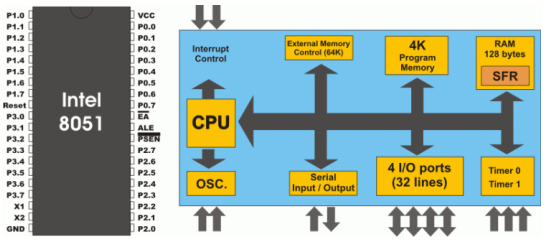
Pins description (2/4)



- /EA：external access
  - There is no on-chip ROM in 8031 and 8032 .
  - The /EA pin is connected to GND to indicate the code is stored externally.
  - /PSEN & ALE are used for external ROM.
  - For 8051, /EA pin is connected to Vcc.
  - "/" means active low.
- /PSEN：program store enable
  - This is an output pin and is connected to the OE pin of the ROM

La famille des 8051

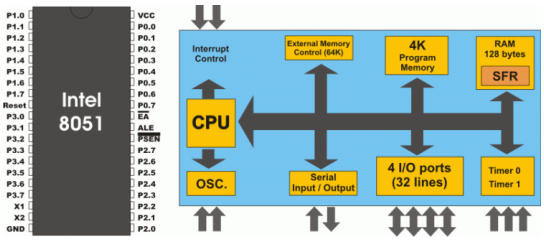
Pins description (3/4)



- ALE：address latch enable
  - It is an output pin and is active high.
  - 8051 port 0 provides both address and data.
  - The ALE pin is used for de-multiplexing the address and data by connecting to the G pin of the 74LS373 latch.
- I/O port pins
  - The four ports P0, P1, P2, and P3.
  - Each port uses 8 pins.
  - All I/O pins are bi-directional.

La famille des 8051

Pins description (4/4)

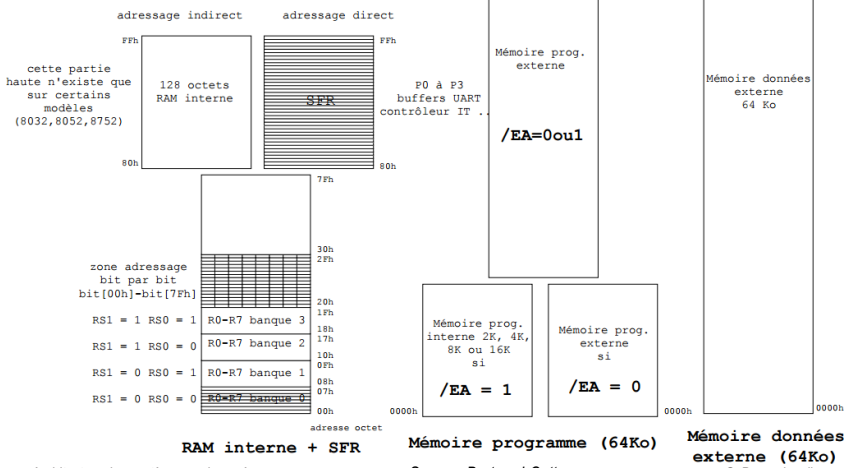


- Port 2 : If there is no intention to use external memory then these port pins are configured as general inputs/outputs. In case external memory is used, the higher address byte, i.e. addresses A8-A15 will appear on this port. Even though memory with capacity of 64Kb is not used, which means that not all eight port bits are used for its addressing, the rest of them are not available as inputs/outputs.
- Port 0 : Similar to P2, if external memory is not used, these pins can be used as general inputs/outputs. Otherwise, P0 is configured as address output (A0-A7) when the ALE pin is driven high (1) or as data output (Data Bus) when the ALE pin is driven low (0).

Source: http://www.mikroe.com/

La famille des 8051

Espaces d'adressage (ROM + RAM)

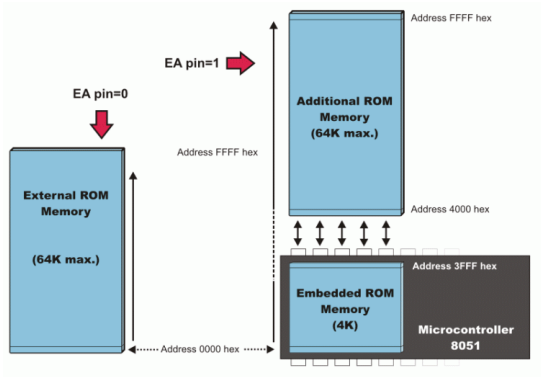


Source: Bertrand Cottenceau

La famille des 8051

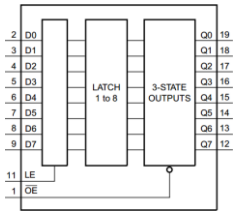
Différents scénarios pour les mémoires… (mémoire ROM)

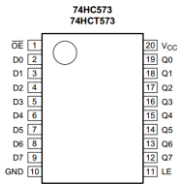
- **EA=0** In this case, the microcontroller completely ignores internal program memory and executes only the program stored in external memory.
- **EA=1** In this case, the microcontroller executes first the program from built-in ROM, then the program stored in external memory.
- In both cases, P0 and P2 are not available for use since being used for data and address transmission. Besides, the ALE and PSEN pins are also used.



Source: http://www.mikroe.com/

Pour comprendre le fonctionnement du prochain slide…





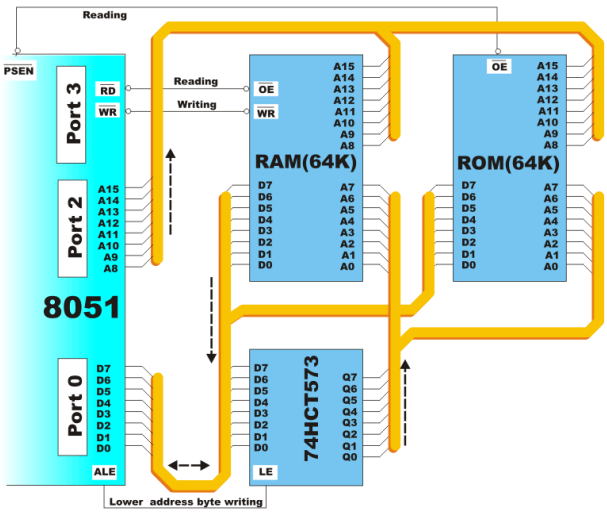
Operating mode	Control		Input	Internal latches	Output
	$\overline{OE}$	LE	Dn		Qn
Enable and read register (transparent mode)	L	H	L	L	L
			H	H	H
Latch and read register	L	L	l	L	L
			h	H	H
Latch register and disable outputs	H	L	l	L	Z
			h	H	Z

[1] H = HIGH voltage level;
h = HIGH voltage level one set-up time prior to the HIGH-to-LOW LE transition;
L = LOW voltage level;
l = LOW voltage level one set-up time prior to the HIGH-to-LOW LE transition;
Z = high-impedance OFF-state.

La famille des 8051

Différents scénarios pour les mémoires… (extensions ROM RAM)

- The whole process described below is performed automatically.
- When the program during execution encounters an instruction which resides in external memory (ROM), the microcontroller will activate its control output ALE and set the first 8 bits of address (A0-A7) on P0. IC circuit 74HCT573 passes the first 8 bits to memory address pins.
- A signal on the ALE pin latches the IC circuit 74HCT573 and immediately afterwards 8 higher bits of address (A8-A15) appear on the port. In this way, a desired location of additional program memory is addressed. It is left over to read its content.
- Port P0 pins are configured as inputs, the PSEN pin is activated and the microcontroller reads from memory chip.
- Similar occurs when it is necessary to read location from external RAM. Addressing is performed in the same way, while read and write are performed via signals appearing on the control outputs RD (is short for read) or WR (is short for write).

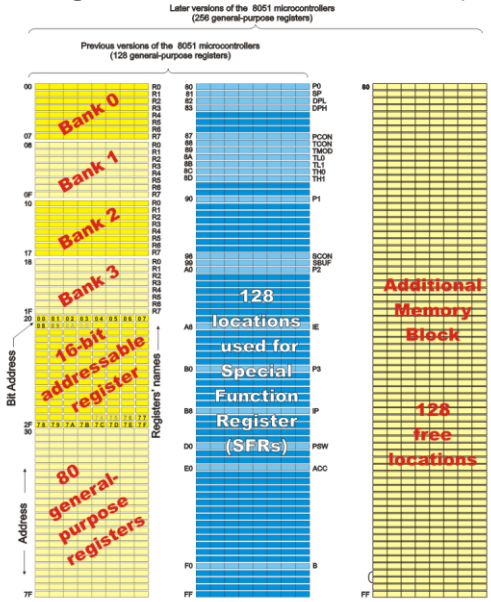


Source: http://www.mikroe.com/

La famille des 8051

Organisation interne (banques de registres, SFRs et accumulateurs)

- The whole process described below is performed automatically.
- When the program during execution encounters an instruction which resides in external memory (ROM), the microcontroller will activate its control output ALE and set the first 8 bits of address (A0-A7) on P0. IC circuit 74HCT573 passes the first 8 bits to memory address pins.
- A signal on the ALE pin latches the IC circuit 74HCT573 and immediately afterwards 8 higher bits of address (A8-A15) appear on the port. In this way, a desired location of additional program memory is addressed. It is left over to read its content.
- Port P0 pins are configured as inputs, the PSEN pin is activated and the microcontroller reads from memory chip.
- Similar occurs when it is necessary to read location from external RAM. Addressing is performed in the same way, while read and write are performed via signals appearing on the control outputs RD (is short for read) or WR (is short for write).



Source: http://www.mikroe.com/

La famille des 8051

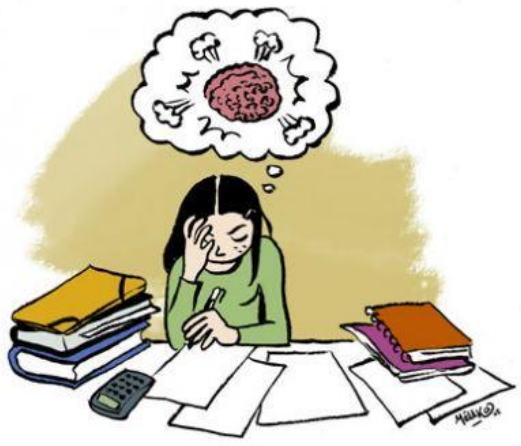
Exemple de références actuelles de 2 fondeurs

- www.atmel.com
  - Remarque: 8051-1C vs 8051-12C:

> The Atmel® AT89LP family takes 8051 microcontroller power to a new level. These high performance 8-bit microcontrollers execute most instructions in a single clock cycle, compared to 12 clock cycles in the classic 8051 CPU. At the same MIPS throughput as the classic 8051, existing applications can use a much lower clock frequency, enabling power consumption reduction by up to 80 percent. Application performance can be boosted up to 20 MIPS throughput 12 times faster than the traditional 8051 core.

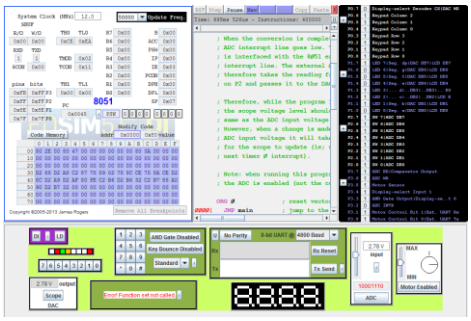
- www.silabs.com

Exercice



Emulateur 8051

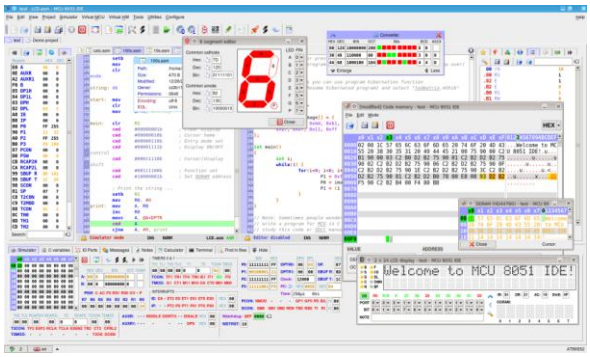
- Téléchargez edsim51di sur la page http://www.edsim51.com/
- Exécutez le
- Etudiez le schéma électronique de référence
- Chargez le programme d'exemple adcToDac.asm
- Etudiez le fonctionnement de ce programme d'exemple



La famille des 8051

Outils Open Source pour le 8051

- http://www.moravia-microsystems.com/mcu-8051-ide/
  - Un autre émulateur (supportant le langage C)



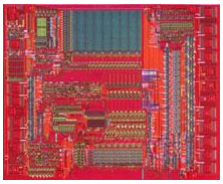
- http://sdcc.sourceforge.net/
  - SDCC - Small Device C Compiler

La grande famille des PIC

Présentation

MICROCHIP

- Famille des « PICmicro » de Microchip
- Dérivés du PIC1650 développé en 1977 par General Instrument (scindée ensuite en plusieurs entreprises)
- Pourrait être traduit en:
  - « Peripheral Interface Controller »
  - « Programmable Intelligent Computer »
  - « Programmable Integrated Circuit »



- En 2009, Microchip revendiquait avoir vendu plus de **6 milliards** de microcontrôleurs PIC

La grande famille des PIC

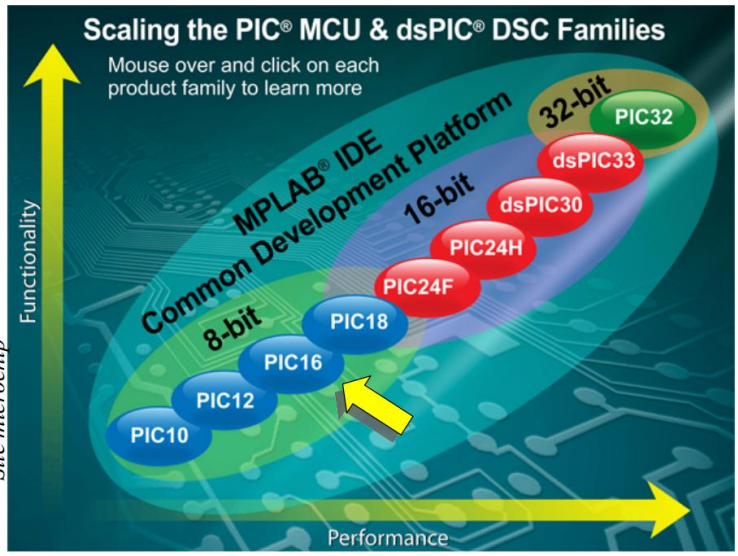
Familles de PIC

- **8 bits**
  - **10F 12F** : ce sont des PIC minuscules (environ 6 pattes) qui possèdent quelques fonctions essentielles (timer, comparateur ou convertisseur analogique/numérique).
  - **16F** : ce sont les PIC les plus répandus. Parmi leurs caractéristiques principales, nous pouvons citer : architecture 8 bits, timers, compteurs, modules PWM, convertisseur analogique/numérique, modules de communication, nombreux ports d'E/S…
  - **18F** : Ces PIC sont assez semblables aux 16F.Ils sont optimisés pour la programmation en langage C, grâce à un plus grand nombre d'instructions assembleur. Ils tendent de plus en plus à remplacer les 16F.
- **16 bits**
  - **PIC24** : Tout en restant dans le même type d'application que les 18F, les PIC24 offrent de bien meilleures performances grâce à leur architecture 16 bits, tout en conservant un grand nombre de périphériques.
  - **dsPIC** : En combinant architecture 16 bits, cœur de calcul DSP et périphériques plus performants et plus variés, les dsPIC sont le choix idéal pour le contrôle moteur complexe, le traitement du signal…
- **32 bits**
  - **PIC32** : Ce sont aujourd'hui les produits les plus évolués de la gamme Microchip. Leur utilisation est réservée à des applications complexes et gourmandes

 Source: http://clubelek.insa-lyon.fr/

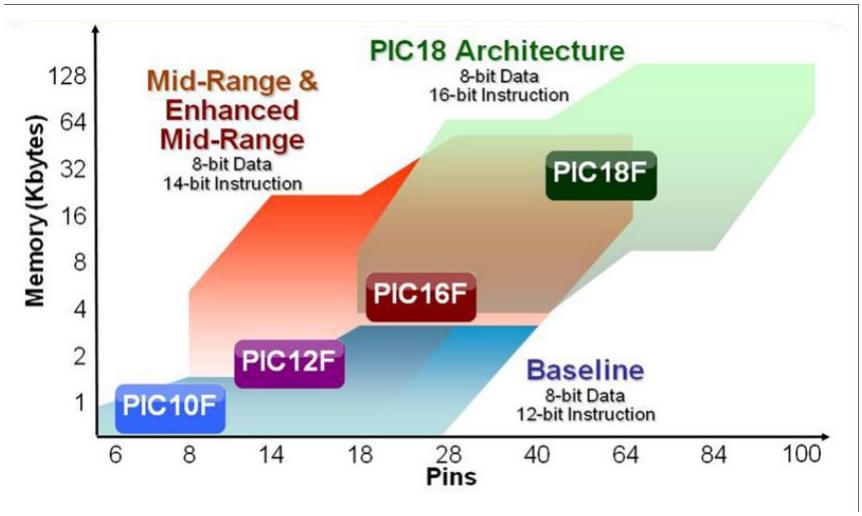
La grande famille des PIC

Familles de PIC



La grande famille des PIC

Familles de PIC



La grande famille des PIC

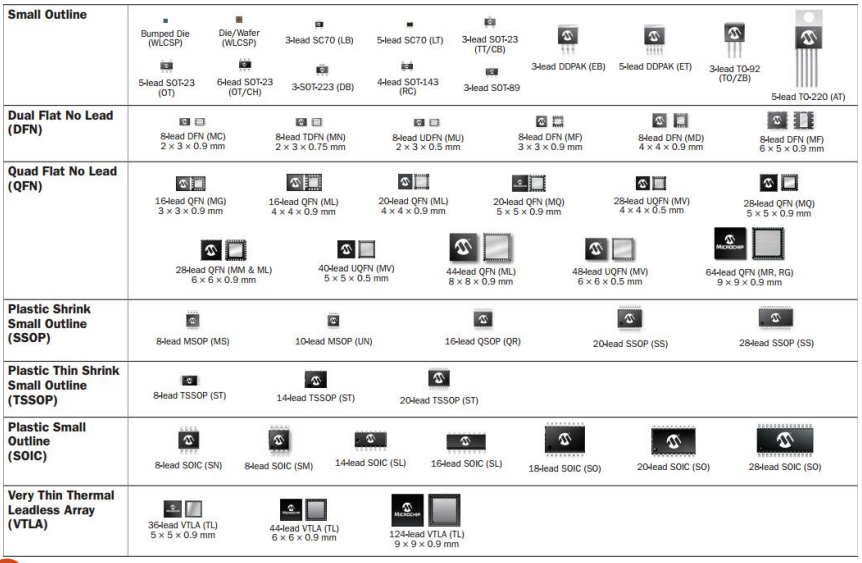
Nomenclature

- Les modèles de PIC courants sont repérés par une référence de la forme :
  - 2 chiffres : famille du PIC (10, 12, 16, 18, 24 ,32) ou dsPIC (30, 33) — 2 familles très rares ont été également introduites PIC14 / PIC17.
  - 1 lettre : type de mémoire de programme (C ou F). Le F indique en général qu'il s'agit d'une mémoire flash et donc effaçable électriquement. La lettre C indique en général que la mémoire ne peut être effacée que par exposition aux ultra-violets (exception pour le PIC16C84 qui utilise une mémoire EEPROM donc effaçable électriquement). Un L peut être ajouté devant pour indiquer qu'il s'agit d'un modèle basse tension (exemple : 2 V à 5,5 V si LF — 4,2 V à5,5 V si F).
  - un nombre de 2 à 4 chiffres : modèle du PIC au sein de la famille.
  - un groupe de lettres pour indiquer le boîtier et la gamme de température.
- Par exemple, le PIC18LF4682-I/P est un microcontrôleur de la famille PIC18, basse tension (L), à mémoire flash (F), modèle 4682, gamme de température industrielle (I) et boîtier DIL40.

(Toutefois il y a maintenant des exceptions : PIC18F25K20 ou PIC18F96J60 par exemple)

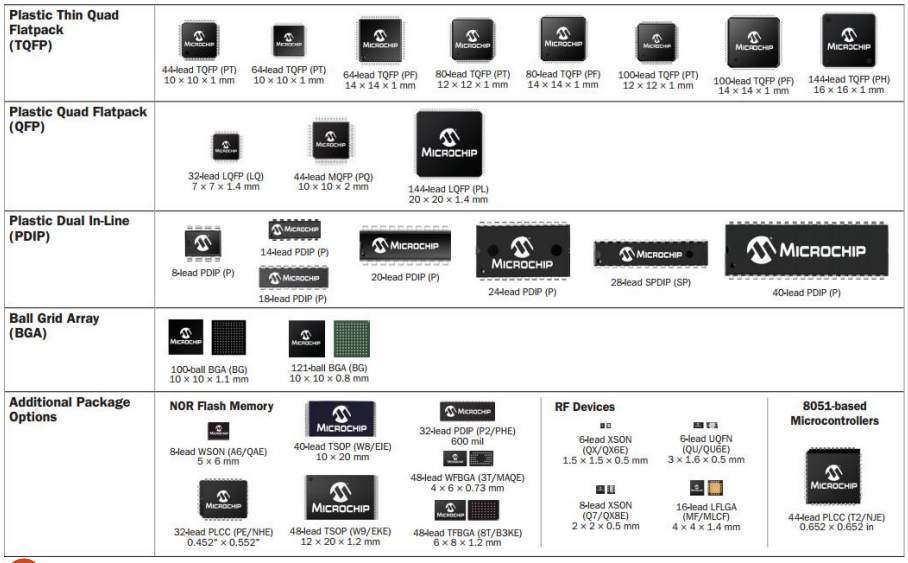
La grande famille des PIC

Il y en a pour tous les goûts…



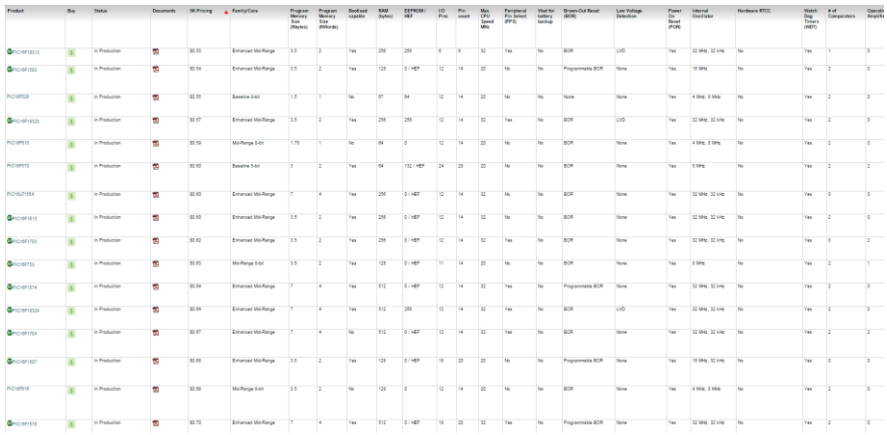
La grande famille des PIC

Il y en a pour tous les goûts…



La grande famille des PIC

De ~~nombreuses~~ références… très nombreuses !!!



→ Voir le site de microchip

La grande famille des PIC

Focus sur la famille des 8 bits

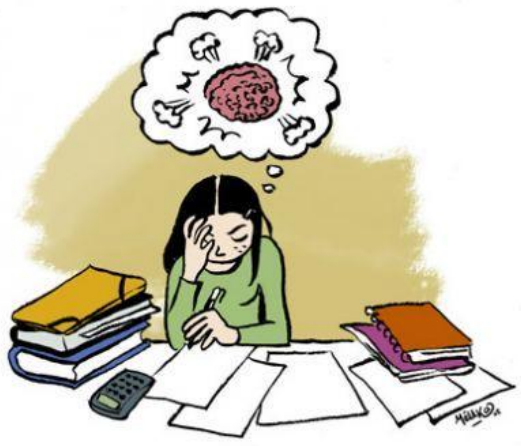
	Baseline Architecture	Mid-Range Architecture	Enhanced Mid-Range Architecture	PIC18 Architecture
Pin Count	6-40	8-64	8-64	18-100
Interrupts	No	Single interrupt capability	Single interrupt capability with hardware context save	Multiple interrupt capability with hardware context save
Performance	5 MIPS	5 MIPS	8 MIPS	Up to 16 MIPS
Instructions	33, 12-bit	35, 14-bit	49, 14-bit	83, 16-bit
Program Memory	Up to 3 KB	Up to 14 KB	Up to 28 KB	Up to 128 KB
Data Memory	Up to 138 Bytes	Up to 368 Bytes	Up to 1,5 KB	Up to 4 KB
Hardware Stack	2 level	8 level	16 level	32 level
Features	▪ Comparator ▪ 8-bit ADC ▪ Data Memory ▪ Internal Oscillator	In addition to Baseline: ▪ SPI/I²C™ ▪ UART ▪ PWMs ▪ LCD ▪ 10-bit ADC ▪ Op Amp	In addition to Mid-Range: ▪ Multiple Communication Peripherals ▪ Linear Programming Space ▪ PWMs with Independent Time Base	In addition to Enhanced Mid-Range: ▪ 8x8 Hardware Multiplier ▪ CAN ▪ CTMU ▪ USB ▪ Ethernet ▪ 12-bit ADC
Highlights	Lowest cost in the smallest form factor	Optimal cost to performance ratio	Cost effective with more performance and memory	High performance, optimized for C programming, advanced peripherals
Total Number of Devices	16	58	29	193
Families	PIC10, PIC12, PIC16	PIC12, PIC16	PIC12FXXX, PIC16F1XX	PIC18

La grande famille des PIC

Focus sur la famille des 8 bits

	Baseline Architecture	Mid-Range Architecture	Enhanced Mid-Range Architecture	PIC18 Architecture
Pin Count	6-40	8-64	8-64	18-100
Interrupts	No	Single interrupt capability	Single interrupt capability with hardware context save	Multiple interrupt capability with hardware context save
Performance	5 MIPS	5 MIPS	8 MIPS	Up to 16 MIPS
Instructions	33, 12-bit	35, 14-bit	49, 14-bit	83, 16-bit
Program Memory	Up to 3 KB	Up to 14 KB	Up to 28 KB	Up to 128 KB
Data Memory	Up to 138 Bytes	Up to 368 Bytes	Up to 1,5 KB	Up to 4 KB
Hardware Stack	2 level	8 level	16 level	32 level
Features	▪ Comparator ▪ 8-bit ADC ▪ Data Memory ▪ Internal Oscillator	In addition to Baseline: ▪ SPI/I²C™ ▪ UART ▪ PWMs ▪ LCD ▪ 10-bit ADC ▪ Op Amp	In addition to Mid-Range: ▪ Multiple Communication Peripherals ▪ Linear Programming Space ▪ PWMs with Independent Time Base	In addition to Enhanced Mid-Range: ▪ 8x8 Hardware Multiplier ▪ CAN ▪ CTMU ▪ USB ▪ Ethernet ▪ 12-bit ADC
Highlights	Lowest cost in the smallest form factor	Optimal cost to performance ratio	Cost effective with more performance and memory	High performance, optimized for C programming, advanced peripherals
Total Number of Devices	16	58	29	193
Families	PIC10, PIC12, PIC16	PIC12, PIC16	PIC12FXXX, PIC16F1XX	PIC18

Exercice

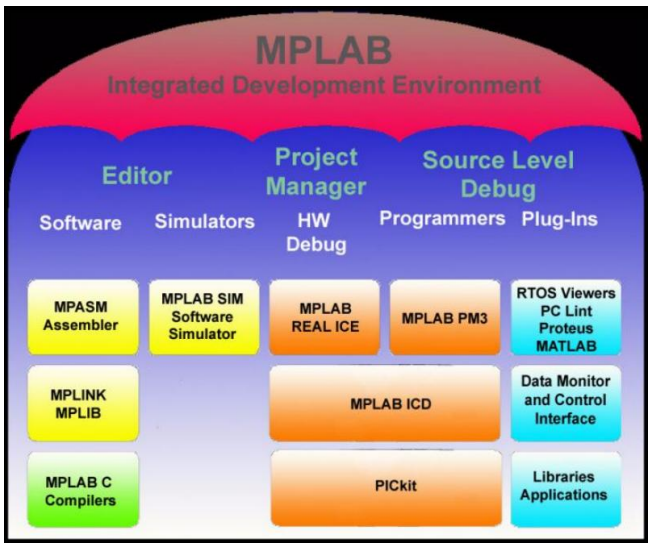


Outil « Microchip Advanced Part Selector »

- Ouvrez la page http://www.microchip.com/maps/microcontroller.aspx
- Filtrez les PIC ayant les caractéristiques suivantes:
  - Références courantes
  - 8 bits
  - Au moins un port CAN et un port SPI
  - 40 broches
- Comparer les caractéristiques du premier et du dernier de la liste proposée
- Réitérer en sélectionnant les caractéristiques suivantes:
  - MCU 16 bits
  - 1 port USB Full Speed Device / Host / OTG
  - 2 ports CAN

La grande famille des PIC

Outils de développement



 Source: http://electroniciens.dr14.cnrs.fr/

La grande famille des PIC

Outils de développement

Logiciel gratuit inclus dans MPLAB IDE	
Programmer's Text Editor	Utilisez l'éditeur pour écrire du code source et le sauvegarde. Le code source peut être destiné à l'assembleur ou aux compilateurs C.
Project Manager	Permet de grouper à la source des fichiers, puis de les envoyés vers les outils logiciels pour l'assemblage et / ou la compilation.
MPASM™/MPLINK™	Le code écrit dans MPLAB est transformée en "code machine" par l'assembleur en une suite de un et de zéro, qui seront ensuite programmé dans un MCU. L'éditeur de liens prend soin de mettre le code en mémoire, d'attribuer des registres pour les variables, et de s'assurer que chaque module source (généralement écrites dans un fichier texte) peuvent accéder à des fonctions et des variables à partir de modules d'autres sources.
MPLAB SIM	Ceci est le simulateur qui est intégré à MPLAB IDE. Il prend en charge le débogage des PIC MCU et dsPIC DSC. Il utilise votre PC pour simuler les instructions MCU et simule également de nombreuses fonctions des périphériques, y compris la conversion A/D, l'USARTs, et les broches I/O. Des signaux de stimulus peuvent être injectés dans les registres ou sur des broches et des journaux de changements de registre peuvent être sauvegardés dans un fichier texte pour une analyse ultérieure.

Source: http://electroniciens.dr14.cnrs.fr/

La grande famille des PIC

Outils de développement

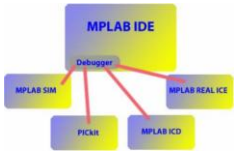
Logiciels en option pour MPLAB IDE	
MPLAB C Compilers	Compilateurs C conforme à la norme ANSI pour les microcontrôleurs PIC18, PIC24 et PIC32 et dsPIC DSC. C'est une application 32 bits, utilisable avec la console Windows, mais est également entièrement intégré dans MPLAB IDE. Les projets, les options du compilateur et les personnalisations de l'éditeur de liens peut entièrement s'effectuer dans MPLAB IDE, fournissant une interface graphique pour ce puissant compilateur. Nodification des erreurs et basculer instantanément vers des lignes correspondantes dans le code source, des fenêtres de visualisation permettent de voir la structure des données avec les types de données définis, y compris en virgule flottante.

Programmateurs optionnels à MPLAB IDE	
PICSTART Plus	Un programmeur de périphérique bas-coût, soutenant principalement PIC dispositifs DIP.
MPLAB PM3	Un programmeur universel supporter la plupart des PIC et des dispositifs dsPIC et forfaits.
MPLAB ICD	Un débogueur à faible coût / programmeur qui travaille avec de nombreux dernière version de Flash microcontrôleurs PIC et dsPIC DSC.
PICkit développement Programmeur / Debugger	Permet le débogage in-circuit et la programmation des microcontrôleurs PIC ® sélectionné.

Source: http://electroniciens.dr14.cnrs.fr/

La grande famille des PIC

Outils de développement



Microchip In-Circuit Emulators and Debuggers	
MPLAB REAL ICE	MPLAB REAL ICE est la prochaine génération d'émulateur de Microchip à grande vitesse pour les microcontrôleurs. Il débogue des programmes de microcontrôleurs PIC et dsPIC avec l'interface utilisateur graphique puissante de l'environnement de développement intégré MPLAB IDE.
MPLAB ICD	Certains Flash-based microcontrôleurs PIC et dsPIC DSC ont la logique supplémentaire sur chaque puce qui permet à un débogueur en circuit pour télécharger du code dans la puce, puis de l'exécuter. La logique supplémentaire permet à l'application du firmware de s'arrêter à des points d'arrêt, où l'ingénieur peut voir les registres internes et vérifier l'état de la puce au point d'arrêt. Le code peut être exécuté en mode pas à pas, les variables et les registres peuvent être modifiés afin de déterminer si le code fonctionne correctement. La mémoire programme du microcontrôleur Flash cible est utilisée pendant le débogage. Un circuit de debug permet l'interconnexion même après que le produit soit entré en production, il peut être débogué et reprogrammé après coup.
PICkit Development Programmer/Debugger	Enables in-circuit debugging and programming of selected PIC® microcontrollers.

La grande famille des PIC

Outils de développement

MPLAB PM3

- RS-232 ou USB
- Intégrée l'interface In-Circuit Serial Programming (ICSP)
  - Mode PC hôte pour un contrôle total
  - Mode sans échec pour sécuriser les données
  - Mode autonome pour la programmation sans PC
- Ligne complète de sockets interchangeables pour soutenir tous la gamme de Microchip et les options de boîtier (vendu séparément)
- Sérialisation SQTP pour la programmation des numéros de série unique.
- Une deuxième ligne de commande DOS interface est disponible pour le contrôle des lots
- Secure Digital (SD) et MultiMedia Card (MMC)

MPLAB ® REAL ICE ™

Bien qu'il s'agisse principalement d'un débogueur, MPLAB REAL ICE sert en tant que programmeur pour les microcontrôleurs Flash. MPLAB REAL ICE, utilise le dispositif de Microchip In-Circuit Serial Programming et tant que les trois broches peuvent être correctement conduites aux microcontrôleurs PIC et dsPIC DSC, le firmware peut être reprogrammation après la fabrication du produit de l'application.

La grande famille des PIC

Outils de développement

MPLAB ICD

Aussi et surtout un débogueur, MPLAB ICD permet de programmer plusieurs microcontrôleurs Flash. MPLAB ICD utilise la technologie de Microchip In-Circuit Serial Programming.
MPLAB ICD peut également être utilisé pour tester et déboguer une application finale, utilisant une interface simple. De nombreux concepteurs laisser un connecteur ICD sur l'application finale (ou au moins une place sur le PCB sur lequel un socket peut être soudé si nécessaire) à utiliser pour le débogage sur le terrain, faire des modifications du firmware, ou la fabrication et les essais en ligne.

PICkit développement Programmeur / Debugger

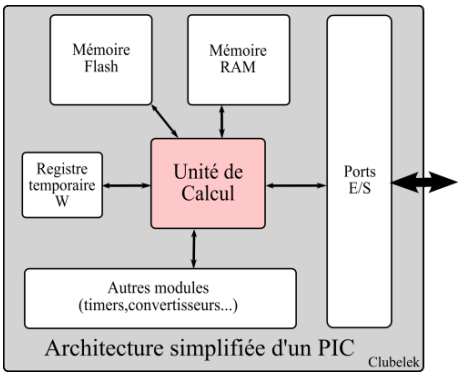
PICkit Debug Express permet le débogage in-circuit sur certains microcontrôleurs PIC. Les fonctions de débogage simple, permettent de stopper le programme pendant le fonctionnement sur le microcontrôleur PIC qui intégré l'application. Lorsque le programme s'arrête sur un point d'arrêt, les registres de fichiers peuvent être examinées et modifiées.

 Source: http://electroniciens.dr14.cnrs.fr/

La grande famille des PIC

Architecture simplifiée

- Architecture Harvard
- RISC
- Temps constant d'exécution des instructions
  (1 ou 2 cycles instruction)



 Source: wikipedia

La grande famille des PIC

Exemple d'un 8 bits « Mid Range »: le PIC 16F84A

MICROCHIP **PIC16F84A**

18-pin *Enhanced* FLASH/EEPROM 8-Bit Microcontroller

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single-cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of Data RAM
- 64 bytes of Data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 Special Function Hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt-on-change
  - Data EEPROM write complete

Peripheral Features:

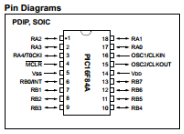
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Special Microcontroller Features:

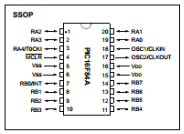
- 10,000 erase/write cycles *Enhanced* FLASH Program memory typical
- 10,000,000 typical erase/write cycles EEPROM Data memory typical
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming™ (ICSP™) - via two pins
- Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Code protection
- Power saving SLEEP mode
- Selectable oscillator options

Pin Diagrams

PDIP, SOIC



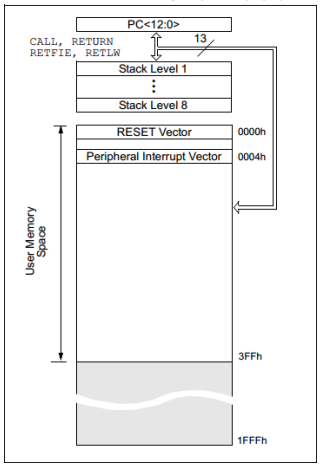
SSOP



CMOS Enhanced FLASH/EEPROM Technology:

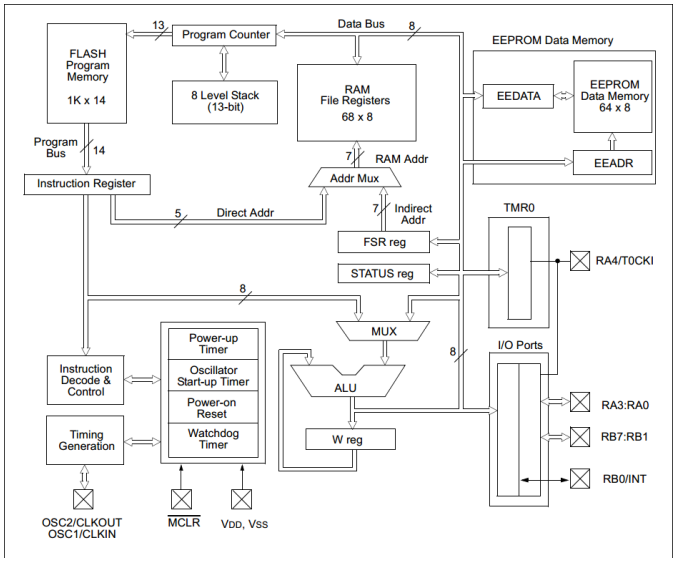
- Low power, high speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 5.5V
  - Industrial: 2.0V to 5.5V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15 µA typical @ 2V, 32 kHz
  - < 0.5 µA typical standby current @ 2V

PROGRAM MEMORY MAP AND STACK - PIC16F84A



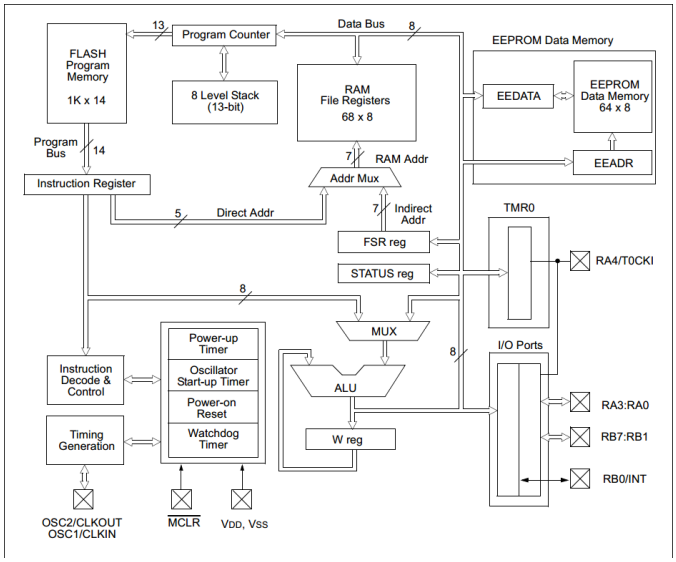
La grande famille des PIC

Exemple d'un 8 bits « Mid Range »: le PIC 16F84A

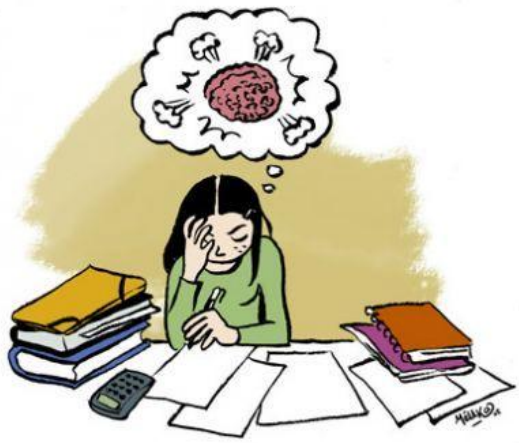


La grande famille des PIC

Exemple d'un 8 bits « Mid Range »: le PIC 16F84A



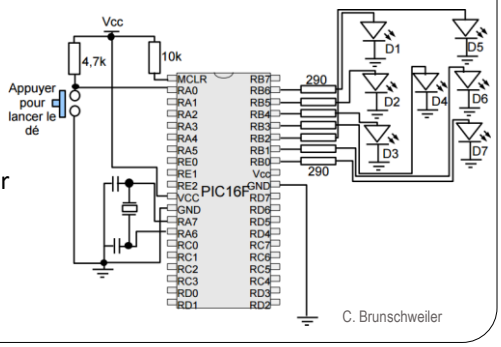
Exercice

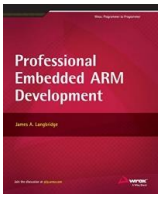


PIC 16F84A

En vous aidant de la datasheet, répondez à ces questions:

- Port d'entrées / sorties
  - Comment faut il procéder pour configurer le bit 0 du port A en entrée et les bits 0 à 6 du port B en sortie?
  - Quelles sont les valeurs à ranger dans un tableau de 6 constantes correspondantes aux 6 combinaisons d'allumage de LED d'un dé électronique (voir schéma).
- Interruptions
  - Expliquez pourquoi l'appui sur le bouton ne peut être géré par une interruption
  - Quelle modification apporter pour corriger cela?





"In the world of embedded systems, you can't work for long without working on an **ARM CPU**."

James a. Langbridge

La **très** grande famille des ARMs

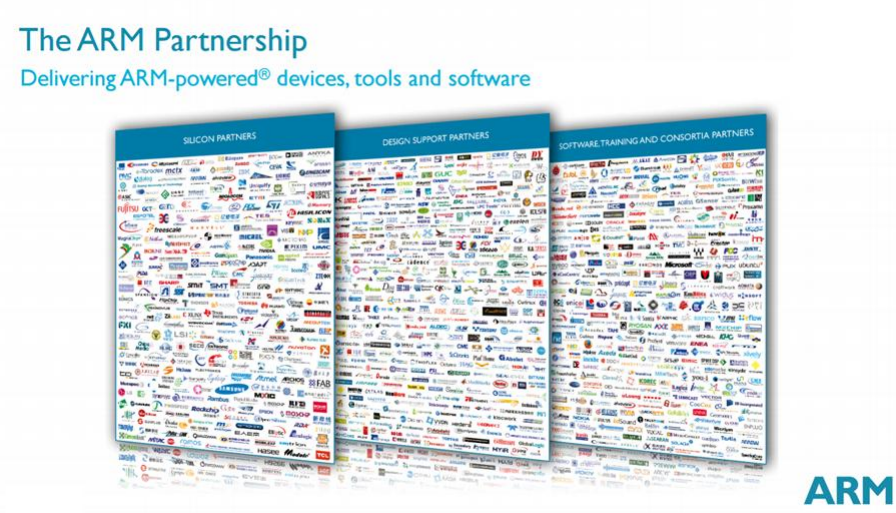
Introduction
Historique
Caractéristiques principales
Quelques processeurs emblématiques à cœur(s) ARM
Quelques produits emblématiques à base de cœur(s) ARM
La famille ARM
Nomenclature
L'architecture ARM
Les jeux d'instructions ARM
Trois petits « extras »

ARM®
The Architecture for the Digital World®

La très grande famille des ARMs
Introduction

- ARM: **A**dvanced **R**ISC **M**achine
- ARM est une joint venture créée entre **Acorn**, **Apple** et **VLSI** en Novembre 1990
- ARM est le **leader** sur le marché des cœurs de micro pour l'embarqué (principalement 32 bits)
- ARM n'est pas un fondeur: il ne fabrique pas (plus) de composants mais fournit, **sous licence**, ses cœurs de CPU aux plus grands fondeurs
- Maison mère basée à Cambridge, UK
- Design centers à Austin (Texas), Cambridge, Sophia Antipolis et Trondheim (Norvège / GPU)
- Principalement connue pour ses cœurs de micros, ARM est également:
  - un éditeur d'outils de développement (marques **Realview** et **Keil**)
  - un concepteur de processeurs graphiques (série **MALI**)

La très grande famille des ARMs
Introduction



La très grande famille des ARMs

Introduction

La très grande famille des ARMs

Historique

- Historiquement, ARM: **A**corn **R**ISC **M**achine
- Acorn est une société anglaise (basée à Cambridge) qui, à la fin des années 70 a conçu et fabriqué un processeur 8 bits, le 6502, sur lequel la société base un ensemble de micro-ordinateurs.
- Le plus emblématique est le « **BBC micro** », vendu à 1,5 millions d'exemplaires (85% des ventes d'ordinateurs aux écoles anglaises).

- Au milieu des années 80, Acorn se lance dans le développement d'une nouvelle génération de processeur basée sur la technologie RISC.
- En avril 1985, après seulement 18 mois de travaux, l'ARM 1 est le 1er (co)processeur RISC à être commercialisé
- En 1987, le premier ordinateur à base de cœur ARM est l'Acorn Archimedes (ARM 2 puis ARM 3)

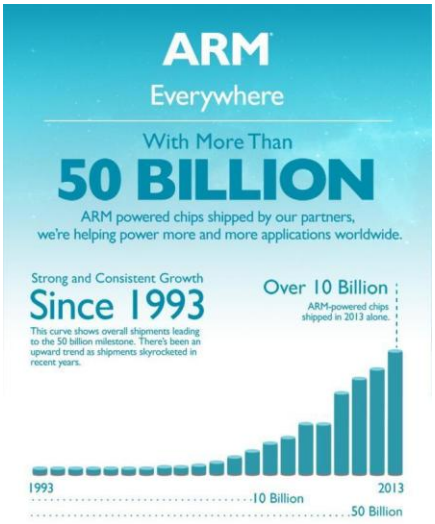
La très grande famille des ARMs

Historique

- Là où Apple, Atari et Commodore, avec un Motorola 68000 à 8 Mhz ne disposait que d'une puissance de 1 MIPS, Acorn disposait lui d'une puissance de 4,5 MIPS à même fréquence avec son processeur maison.
- En novembre 1990, face à l'intérêt d'Apple vis-à-vis de l'architecture des processeurs ARM, la joint venture ARM est créée entre Appel, Acorn et VLSI
- En 1994, le RISC PC (ARM610, ARM700 puis ARM 710 et 710a) succède à l'Acorn Archimedes (équipé du RISC OS).

La très grande famille des ARMs

Historique



La très grande famille des ARMs

Caractéristiques principales

- Processeurs 32 bits (et 64 bits pour les plus récents) à organisation **Von Neumann** (principalement)
- Nombre important de registres (R0 à R16)
- Architecture « **Load and Store** »
  - Les opérations sur les données ne se font qu'à partir du contenu des registres
- Instructions uniformes et de largeur fixe
- La plupart des instructions peuvent être exécutées sous condition
- La plupart des cœurs ARM ont 2 sets d'instructions
  - 32-bit **ARM** Instruction Set
  - 16/32-bit **Thumb** Instruction Set
- Bon ratio vitesse / consommation
- Haute densité de code

La très grande famille des ARMs

Caractéristiques principales

- La plupart des instructions ne prennent qu'un cycle d'horloge
- Vitesse allant de 1 MHz à 1,25 GHz
- Certains cœurs ARM supportent JAVA **Jazelle** DBX (Direct Byte code eXecution)
- DSP Enhanced Instructions
- Support de la technologie **TrustZone** (sécurité)
- **In Circuit Debugger**

- A noter:
  - **Word** = 32 bits
  - **Halfword** = 16 bits
  - **Doubleword** = 64 bits

La très grande famille des ARMs

Quelques processeurs emblématiques à cœur(s) ARM

- Famille **AT91** d'Atmel (SAM9 / SAM7)
- Famille **LPC** de NXP
- Famille **OMAP** de Texas Instrument
- Famille **iMX** de Freescale
- Famille **STM32** de ST Microelectronics
- Famille **Tegra** de Nvidia
- Famille **Snapdragon** de Qualcomm
- Famille **AXX** de Allwinner
- Famille **Exynos** de Samsung
- Famille **AX** d'Apple
- Famille **RKXXXX** de Rockchip
- Famille **MTXXXX** de Mediatek

La très grande famille des ARMs

Quelques produits emblématiques à base de cœur(s) ARM

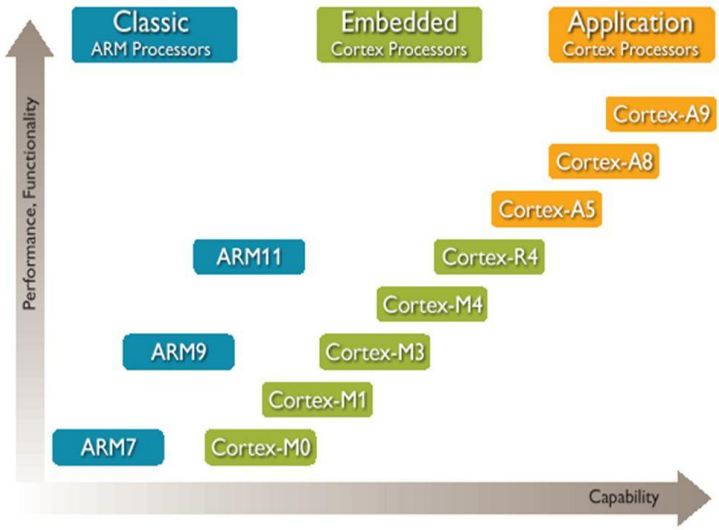
La très grande famille des ARMs

Quelques produits emblématiques à base de cœur(s) ARM

- ARM7TDMI(-S)
  - Game Boy Advance, Nintendo DS, Apple iPod, Lego NXT, GPS Garmin
- ARM920T
  - GPS Garmin, GPS TomTom
- ARM926EJ-S
  - Mobiles Sony séries K et W, NAS de Buffalo et Western Digial
- ARM946E-S
  - Appareils photo Canon
- ARM11 (xxx)
  - Mobiles Nokia, HTC, Samsung, ZTE, iPhone original et 3G, Nintendo 3DS, Radpberry Pi
- Cortex A7
  - Raspberry Pi 2
- Cortex A8
  - iPhone 3GS, HTC Desire, Pandara board, Motorola Droid, Samsung Galaxy S…
- Cortex A9
  - Samsung Galaxy SII, iPhone 4S, Pandaboard, NAS Synology DS215j …
- Cortex M3
  - Arduino Due …

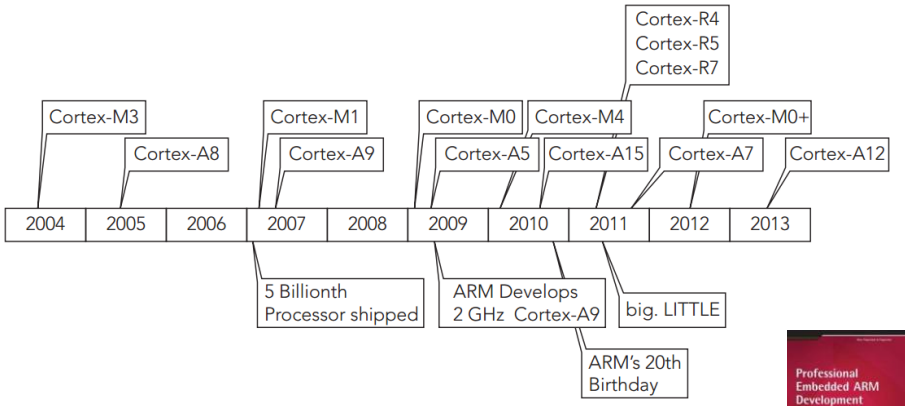
La très grande famille des ARMs

La famille ARM



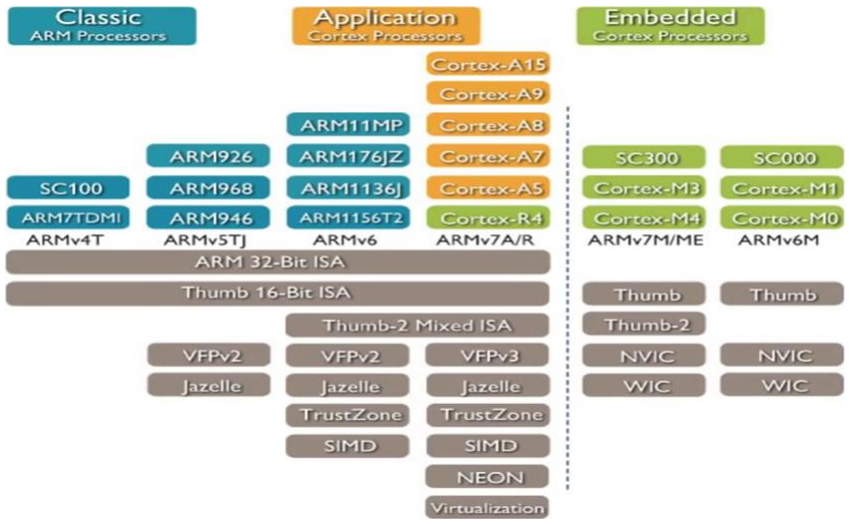
La très grande famille des ARMs

La famille ARM



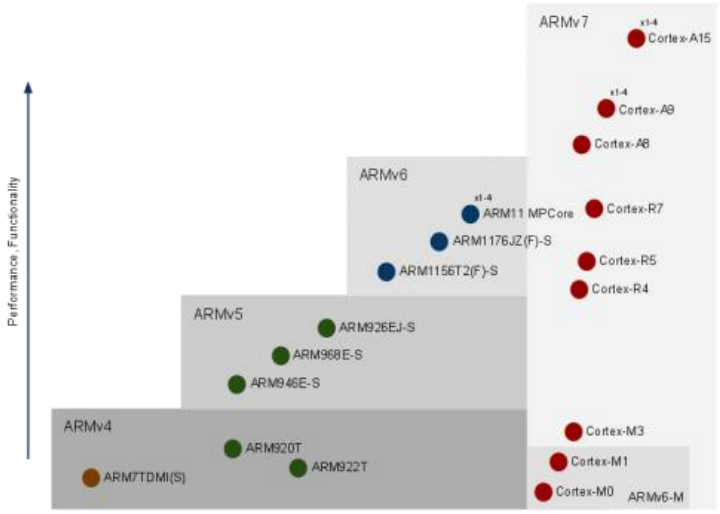
La très grande famille des ARMs

La famille ARM



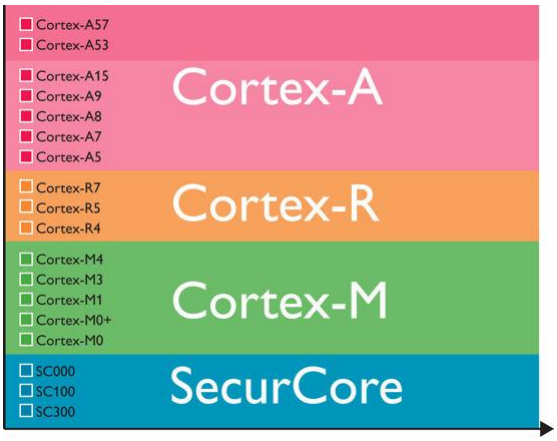
La très grande famille des ARMs

La famille ARM



La très grande famille des ARMs

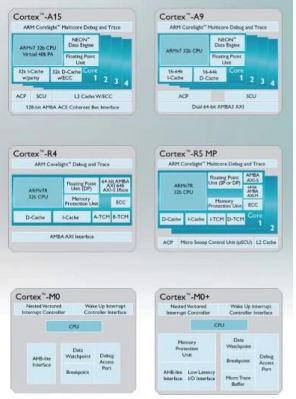
La famille ARM



La très grande famille des ARMs

Profils de l'architecture ARM V7

- **Application profile (ARMv7-A)**
  - Memory management support (MMU)
  - Highest performance at low power
  - Influenced by multi-tasking OS system requirements
  - TrustZone for a safe, extensible system
  - Optional Large Physical Address and Virtualization extensions
- **Real-time profile (ARMv7-R)**
  - Protected memory (MPU)
  - Low latency and predictability 'real-time' needs
  - Tightly Coupled Memories for fast, deterministic access
- **Microcontroller profile (ARMv7-M, ARMv6-M)**
  - Lowest gate count entry point
  - Deterministic and predictable behavior a key priority
  - Deeply embedded use



La très grande famille des ARMs

La famille ARM

Development of the ARM Architecture

v4T

Halfword and signed halfword / byte support
System mode
Thumb instruction set

ARM7TDMI-S

v5TE

Improved ARM/Thumb Interworking
CLZ
Saturated arithmetic
DSP multiply-accumulate instructions

ARM926EJ-S

v6

SIMD Instructions
Multi-processing
v6 Memory architecture
Unaligned data support

Extensions
Thumb-2 (v6T2)
TrustZone (v6Z)
Multicore (v6K)
Thumb only (v6-M)

ARM1136J(F)-S

v7

Thumb-2
NEON
TrustZone
Virtualization

Cortex
Low-Power Leadership from ARM

Architecture Profiles
v7-A (Applications): NEON

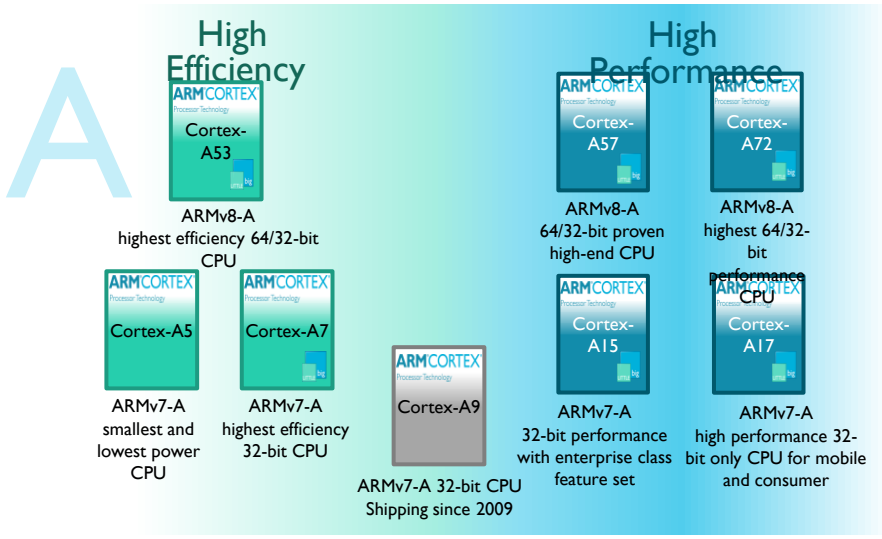
v7-R (Real-time): Hardware divide

v7-M (Microcontroller): Hardware divide, Thumb-2 only

- **Note that implementations of the same architecture can be different:**
  - Cortex-A8 - architecture v7-A with a 13-stage pipeline
  - Cortex-A9 - architecture v7-A with an 8-stage pipeline

ARM® Cortex®-A Current Portfolio

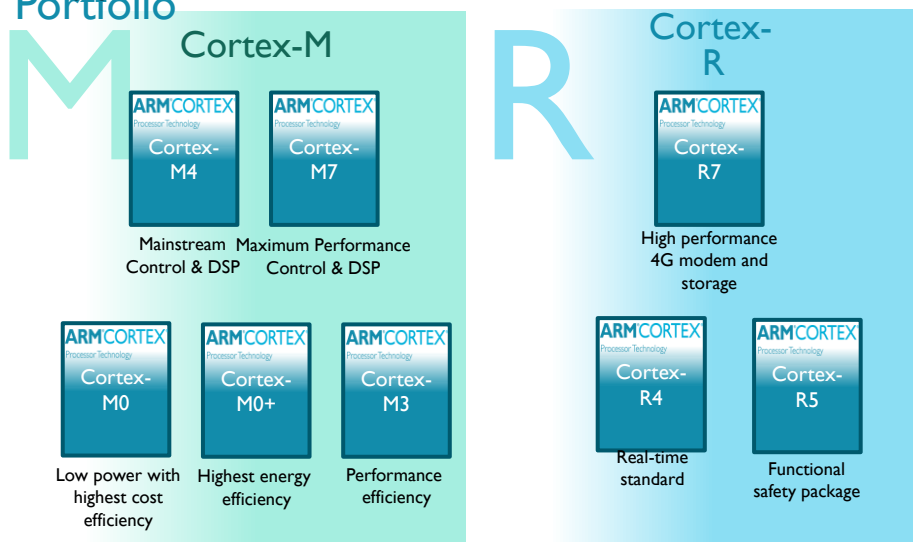
1H 2015



215

ARM® Cortex®-R and Cortex®-M Processor Portfolio

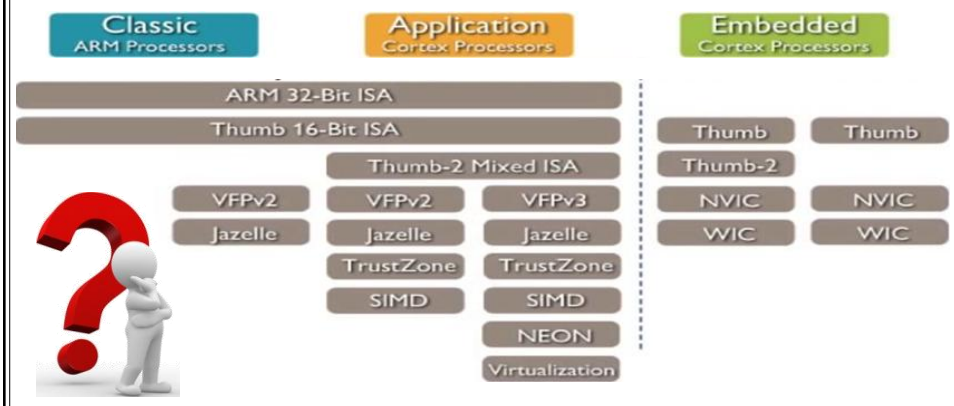
1H 2015



216

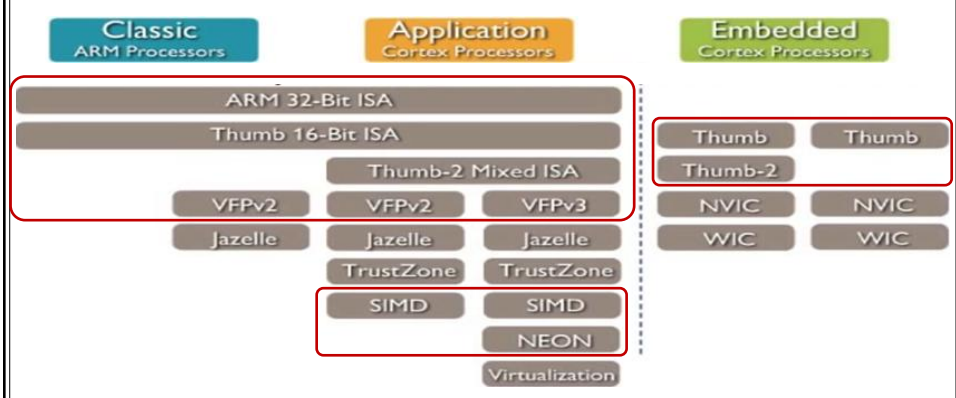
La très grande famille des ARMs

La famille ARM



La très grande famille des ARMs

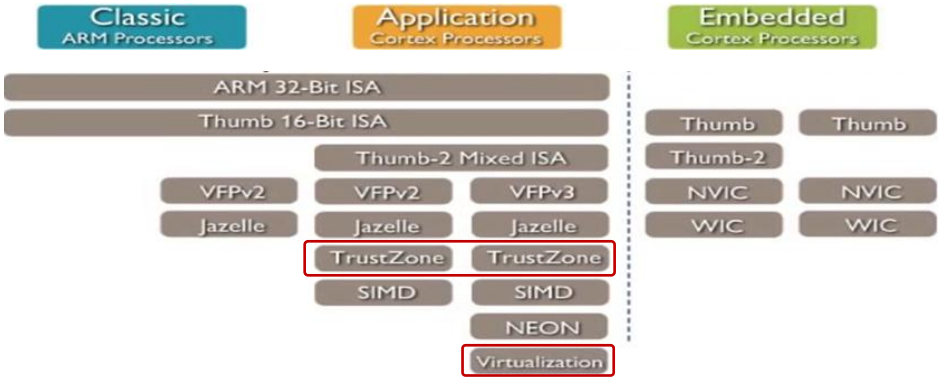
La famille ARM



→ Différents jeux d'instructions ARM

La très grande famille des ARMs

La famille ARM

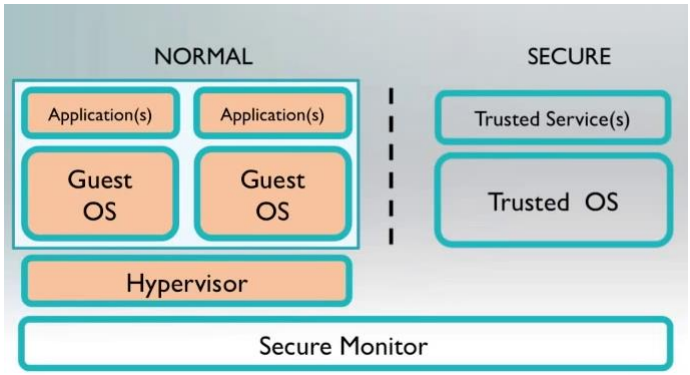


Support pour la sécurité et la sûreté

La très grande famille des ARMs

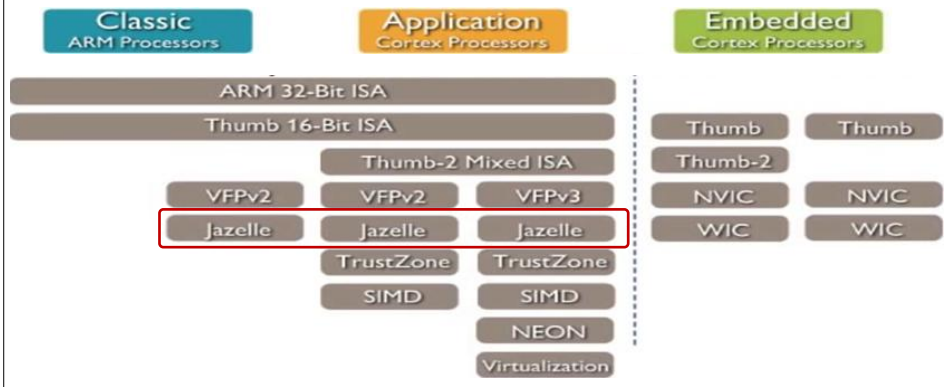
La famille ARM

- TrustZone et Virtualization



La très grande famille des ARMs

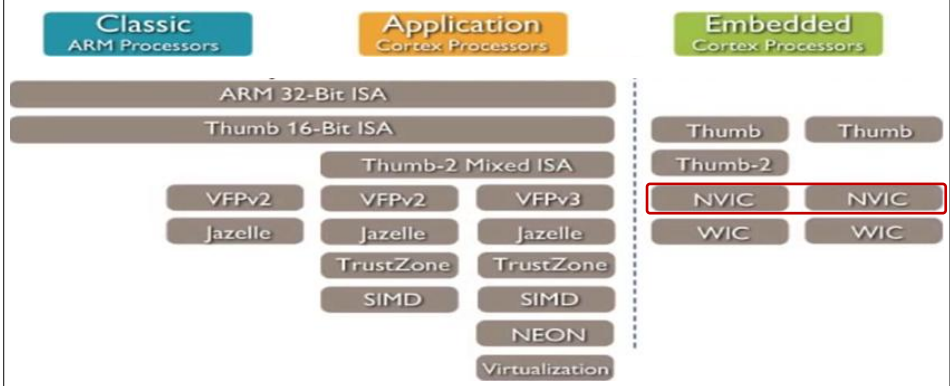
La famille ARM



JAVA Jazelle DBX (Direct Byte code eXecution)

La très grande famille des ARMs

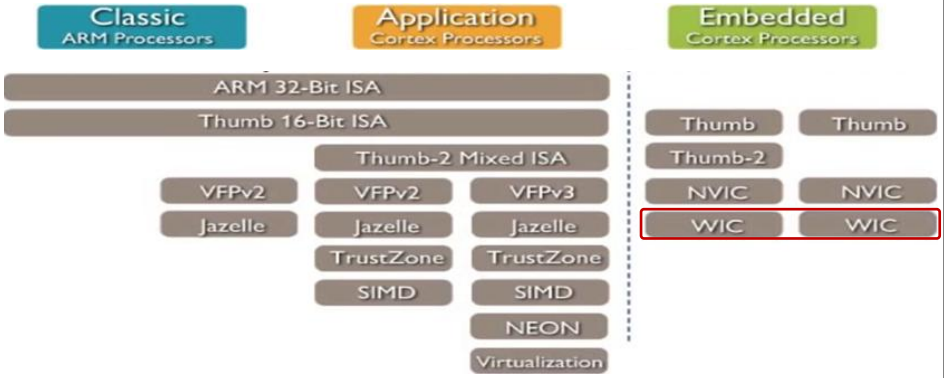
La famille ARM



Nested Vectored Interrupt Controller

La très grande famille des ARMs

La famille ARM



Wakeup Interrupt Controller

La très grande famille des ARMs

Nomenclature

X	Y	Z	DESCRIPTION	EXAMPLE
7			ARM7 core version	ARM7
9			ARM9 core version	ARM9
10			ARM10 core version	ARM10
11			ARM11 core version	ARM11
	1		Cache, write buffer and MMU	ARM710
	2		Cache, write buffer and MMU, Process ID support	ARM920
	3		Physically mapped cache and MMU	ARM1136
	4		Cache, write buffer and MPU	ARM940
	5		Cache, write buffer and MPU, error correcting memory	ARM1156
	6		No cache, write buffer	ARM966
	7		AXI bus, physically mapped cache and MMU	ARM1176
		0	Standard cache size	ARM920
		2	Reduced cache	ARM1022
		6	Tightly Coupled Memory	ARM1156
		8	As for ARM966	ARM968

La très grande famille des ARMs

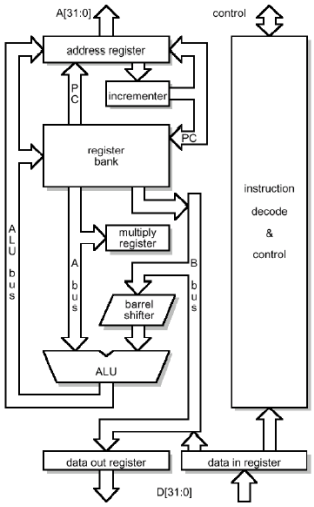
Nomenclature

ATTRIBUTE	DESCRIPTION
D	Supports debugging via the JTAG interface. Automatic for ARMv5 and above.
E	Supports Enhanced DSP instructions. Automatic for ARMv6 and above.
F	Supports hardware floating point via the VFP coprocessor.
I	Supports hardware breakpoints and watchpoints. Automatic for ARMv5 and above.
J	Supports the Jazelle Java acceleration technology.
M	Supports long multiply instructions. Automatic for ARMv5 and above.
T	Supports Thumb instruction set. Automatic for ARMv5 and above.
-S	This processor uses a synthesizable hardware design.

La très grande famille des ARMs

L'architecture ARM

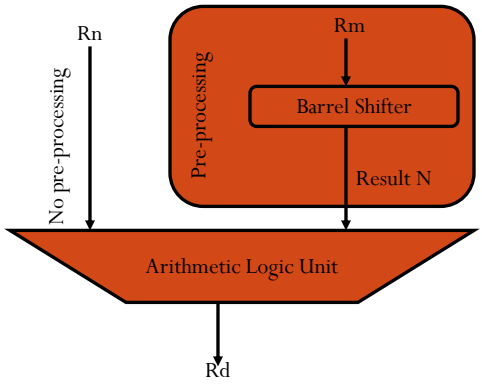
- All data processing instructions can use the barrel shifter (an interesting and unique feature) to shift or rotate an operand
- Can operate in either big or little endian mode
- Memory is byte-oriented; each byte of memory has its own unique address



La très grande famille des ARMs

L'architecture ARM - Barrel Shifter

- Shift the 32 bit binary pattern in one of the source registers left or right by a specific number of positions before it enters the ALU



La très grande famille des ARMs

L'architecture ARM

- *Les 7 modes opératoires des cœurs ARM**

- Most ARM cores have seven basic operating modes
  - Each mode has access to its own stack space and a different subset of registers
  - Some operations can only be carried out in a privileged mode

Mode	Description
Supervisor (SVC)	Entered on reset and when a Supervisor call instruction (SVC) is executed
FIQ	Entered when a high priority (fast) interrupt is raised
IRQ	Entered when a normal priority interrupt is raised
Abort	Used to handle memory access violations
Undef	Used to handle undefined instructions
System	Privileged mode using the same registers as User mode
User	Mode under which most Applications / OS tasks run

Privileged modes

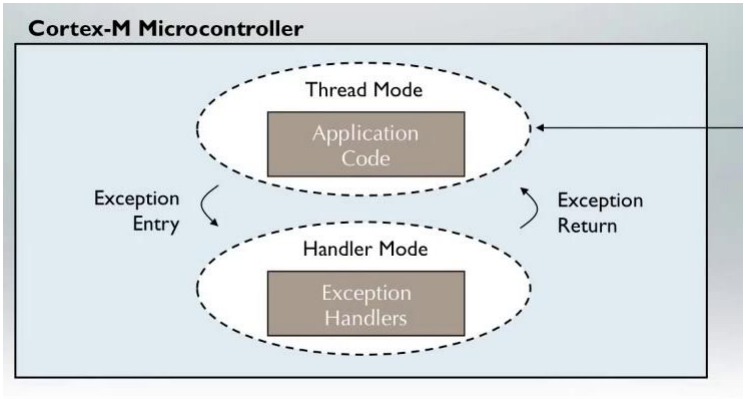
Unprivileged mode

*: non applicable aux Cortex M

La très grande famille des ARMs

L'architecture ARM

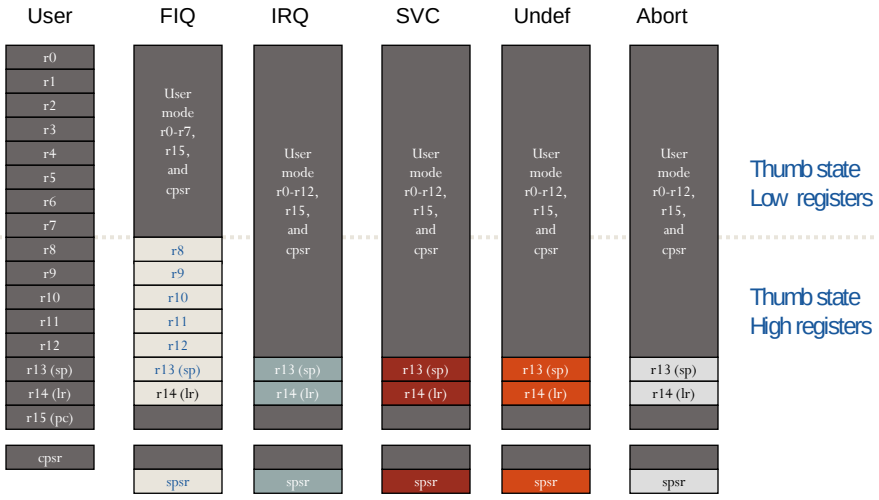
- *Les 2 modes opératoires des cœurs Cortex M**



*: sujets à configuration.

La très grande famille des ARMs

L'architecture ARM - Organisation des registres



Note: System mode uses the User mode register set

La très grande famille des ARMs

L'architecture ARM - Les registres

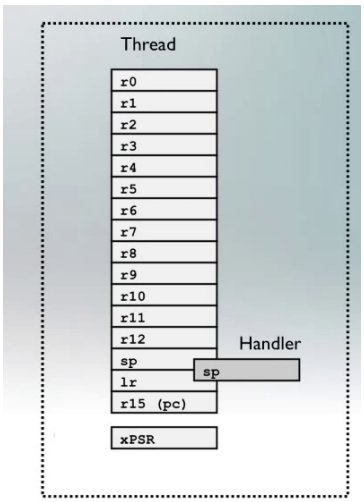
- ARM has 37 registers all of which are 32-bits long.
  - 1 dedicated program counter
  - 1 dedicated current program status register
  - 5 dedicated saved program status registers
  - 30 general purpose registers
- The current processor mode governs which of several banks is accessible. Each mode can access
  - a particular set of r0-r12 registers
  - a particular r13 (the stack pointer, sp) and r14 (the link register, lr)
  - the program counter, r15 (pc)
  - the current program status register, cpsr

  Privileged modes (except System) can also access
  - a particular spsr (saved program status register)

La très grande famille des ARMs

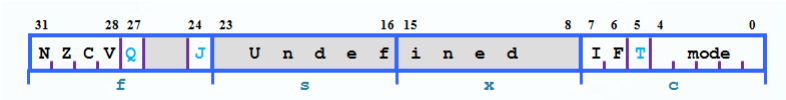
L'architecture ARM - Les registres des Cortex M

- 13 general purpose registers
  - Registers r0 – r7 (Low registers)
  - Registers r8 – r12 (High registers)
- 3 registers with special meaning/usage
  - Stack Pointer (SP) – r13
  - Link Register (LR) – r14
  - Program Counter (PC) – r15
- Special-purpose registers
  - xPSR is the program status register



La très grande famille des ARMs

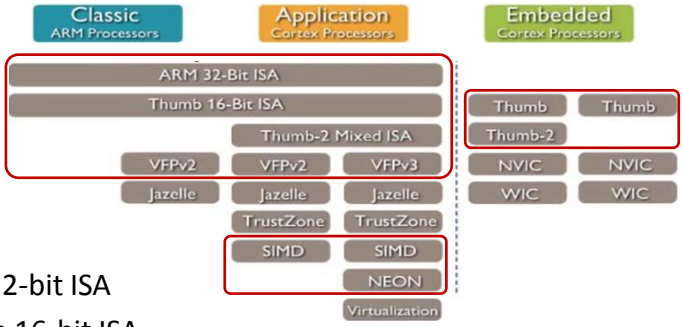
L'architecture ARM - Program Status Registers



- Condition code flags
  - N = **N**egative result from ALU
  - Z = **Z**ero result from ALU
  - C = ALU operation **C**arried out
  - V = ALU operation o**V**erflowed
- Sticky Overflow flag - Q flag
  - Architecture 5TE/J only
  - Indicates if saturation has occurred
- J bit
  - Architecture 5TEJ only
  - J = 1: Processor in Jazelle state
- Interrupt Disable bits.
  - I = 1: Disables the IRQ.
  - F = 1: Disables the FIQ.
- T Bit
  - Architecture xT only
  - T = 0: Processor in ARM state
  - T = 1: Processor in Thumb state
- Mode bits
  - Specify the processor mode

La très grande famille des ARMs

Les jeux d'instructions

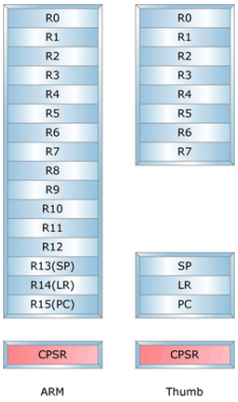


- ARM 32-bit ISA
- Thumb 16-bit ISA
- Thumb 2 Mixed ISA
- VFP (Vector Floating Point)
- SIMD (Single Instruction Multiple Data)
- NEON

La très grande famille des ARMs

Les jeux d'instructions

- Thumb 16-bit ISA
  - Subset of the most commonly used 32-bit ARM instructions.
  - 16 bits long, and have a corresponding 32-bit ARM instruction that has the same effect on the processor model.
  - On execution, 16-bit Thumb instructions are transparently decompressed to full 32-bit ARM instructions in real time, without performance loss.
  - Thumb has all the advantages of a 32-bit core:
    - 32-bit address space
    - 32-bit registers
    - 32-bit shifter, and Arithmetic Logic Unit (ALU)
    - 32-bit memory transfer.
  - Thumb code is typically 65% of the size of ARM code, and provides 160% of the performance of ARM code when running from a 16-bit memory system.



- Thumb 2 Mixed ISA

La très grande famille des ARMs

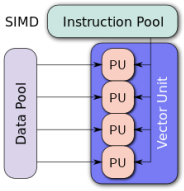
Les jeux d'instructions

- VFP (Vector Floating Point)
  - ARM Floating Point architecture (VFP) provides hardware support for floating point operations in half-, single- and double-precision floating point arithmetic. It is fully IEEE 754 compliant with full software library support.
    - **VFPv1** is obsolete.
    - **VFPv2** is an optional extension to the ARM instruction set in the ARMv5TE, ARMv5TEJ and ARMv6 architectures.
    - **VFPv3** is an optional extension to the ARM, Thumb® and ThumbEE instruction sets in the ARMv7-A and ARMv7-R profiles. VFPv3 implementation is with either thirty-two or sixteen double word registers. The terms VFPv3-D32 and VFPv3-D16 distinguish between these two implementation options. Extending VFPv3 uses the half-precision extensions that provide conversion functions in both directions between half-precision floating-point and single-precision floating-point.

La très grande famille des ARMs

Les jeux d'instructions

- SIMD (Single Instruction Multiple Data)
  - **ARMv6 SIMD Features:**
    - 75% performance increase for audio and video processing
    - Simultaneous computation of 2x16-bit or 4x8-bit operands
    - Fractional arithmetic
    - User definable saturation modes (arbitrary word-width)
    - Dual 16x16 multiply-add/subtract 32x32 fractional MAC
    - Simultaneous 8/16-bit select operations
    - Performance up to 3.2 GOPS at 800MHz
    - Performance is achieved with a "near zero" increase in power consumption on a typical implementation
  - **Applications:**
    - Media streaming
    - Internet appliance
    - MPEG4 and H264 encode/decode
    - Voice and handwriting recognition
    - FFT processing
    - Complex arithmetic

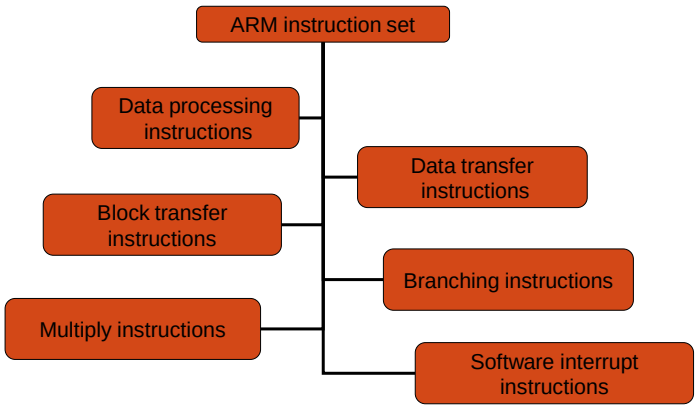


- NEON (Cortex A only)
  - General-purpose SIMD engine
  - At least 2x the performance of ARMv6 SIMD.

NEON™

La très grande famille des ARMs

Les jeux d'instructions - Les différentes instructions



La très grande famille des ARMs

Les jeux d'instructions - Data Processing Instructions

- Classes of data processing instructions
  - Arithmetic operations
  - Bit-wise logical operations
  - Register-movement operations
  - Comparison operations
- Operands: 32-bits wide;
  there are 3 ways to specify operands
  - come from registers
  - the second operand may be a constant (immediate)
  - shifted register operand
- Result: 32-bits wide, placed in a register
  - long multiply produces a 64-bit result

239

Source: Aleksandar Milenkovic

La très grande famille des ARMs

Les jeux d'instructions - Data Processing Instructions

Arithmetic Operations

ADD r0, r1, r2	r0 := r1 + r2
ADC r0, r1, r2	r0 := r1 + r2 + C
SUB r0, r1, r2	r0 := r1 - r2
SBC r0, r1, r2	r0 := r1 - r2 + C - 1
RSB r0, r1, r2	r0 := r2 - r1
RSC r0, r1, r2	r0 := r2 - r1 + C - 1

Bit-wise Logical Operations

AND r0, r1, r2	r0 := r1 and r2
ORR r0, r1, r2	r0 := r1 or r2
EOR r0, r1, r2	r0 := r1 xor r2
BIC r0, r1, r2	r0 := r1 and (not) r2

Register Movement

MOV r0, r2	r0 := r2
MVN r0, r2	r0 := not r2

Comparison Operations

CMP r1, r2	set cc on r1 - r2
CMN r1, r2	set cc on r1 + r2
TST r1, r2	set cc on r1 and r2
TEQ r1, r2	set cc on r1 xor r2

240

Source: Aleksandar Milenkovic

La très grande famille des ARMs

Les jeux d'instructions - Data Processing Instructions

- Immediate operands:
  immediate = (0->255) x 2^{2n}, 0 <= n <= 12

ADD r3, r3, #3	r3 := r3 + 3
AND r8, r7, #&ff	r8 := $r7_{[7:0]}$, & for hex

- Shifted register operands
  - the second operand is subject to a shift operation before it is combined with the first operand

ADD r3, r2, r1, LSL #3	r3 := r2 + 8 x r1
ADD r5, r5, r3, LSL r2	r5 := r5 + 2^{r2} x r3

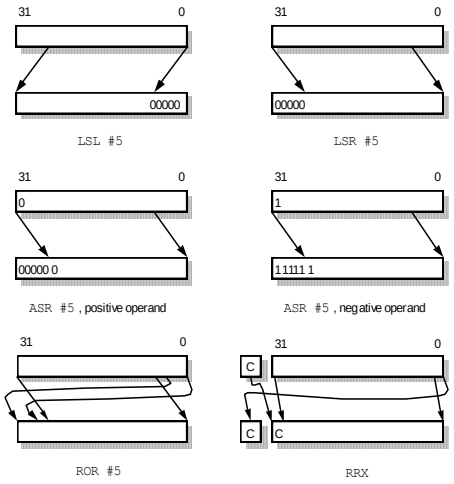
241

Source: Aleksandar Milenkovic

La très grande famille des ARMs

Les jeux d'instructions - Data Processing Instructions

- Shift operations
  - LSL – Logical Shift Left
  - LSR – Logical Shift Right
  - ASR – Arithmetic Shift Right
  - ROR – Rotate Right
  - RRX – Rotate Right Extended by 1 place



242

Source: Aleksandar Milenkovic

La très grande famille des ARMs

Les jeux d'instructions - Data Processing Instructions

- Setting the condition codes
  - Any DPI can set the condition codes (N, Z, V, and C)
    - for all DPIs except the comparison operations a specific request must be made
    - at the assembly language level this request is indicated by adding an `S` to the opcode
    - Example (r3-r2 := r1-r0 + r3-r2)

```
ADDS r2, r2, r0   ; carry out to C
ADC r3, r3, r1    ; … add into high word
```

  - Arithmetic operations set all the flags (N, Z, C, and V)
  - Logical and move operations set N and Z
    - preserve V and either preserve C when there is no shift operation, or set C according to shift operation (fall off bit)

243

Source: Aleksandar Milenkovic

La très grande famille des ARMs

Les jeux d'instructions - Data Processing Instructions

- Multiplies
  - Example (Multiply, Multiply-Accumulate)

MUL r4, r3, r2	r4 := $[r3 \times r2]_{\langle 31:0 \rangle}$
MLA r4, r3, r2, r1	r4 := $[r3 \times r2 + r1]_{\langle 31:0 \rangle}$

  - Note
    - least significant 32-bits are placed in the result register, the rest are ignored
    - immediate second operand is not supported
    - result register must not be the same as the first source register
    - if `S` bit is set the V is preserved and the C is rendered meaningless
  - Example (r0 = r0 x 35)
    - ADD r0, r0, r0, LSL #2 ; r0' = r0 x 5
      RSB r3, r3, r1 ; r0'' = 7 x r0'

244

Source: Aleksandar Milenkovic

La très grande famille des ARMs

Les jeux d'instructions - Data Transfer Instructions

- Single register load and store instructions
  - transfer of a data item (byte, half-word, word) between ARM registers and memory
- Multiple register load and store instructions
  - enable transfer of large quantities of data
  - used for procedure entry and exit, to save/restore workspace registers, to copy blocks of data around memory
- Single register swap instructions
  - allow exchange between a register and memory in one instruction
  - used to implement semaphores to ensure mutual exclusion on accesses to shared data in multis

245

Source: Aleksandar Milenkovic

La très grande famille des ARMs

Les jeux d'instructions - Control Flow Instructions

Branch	Interpretation	Normal uses
B	Unconditional	Always take this branch
BAL	Always	Always take this branch
BEQ	Equal	Comparison equal or zero result
BNE	Not equal	Comparison not equal or non-zero result
BPL	Plus	Result positive or zero
BMI	Minus	Result minus or negative
BCC	Carry clear	Arithmetic operation did not give carry-out
BLO	Lower	Unsigned comparison gave lower
BCS	Carry set	Arithmetic operation gave carry-out
BHS	Higher or same	Unsigned comparison gave higher or same
BVC	Overflow clear	Signed integer operation; no overflow occurred
BVS	Overflow set	Signed integer operation; overflow occurred
BGT	Greater than	Signed integer comparison gave greater than
BGE	Greater or equal	Signed integer comparison gave greater or equal
BLT	Less than	Signed integer comparison gave less than
BLE	Less or equal	Signed integer comparison gave less than or equal
BHI	Higher	Unsigned comparison gave higher
BLS	Lower or same	Unsigned comparison gave lower or same

246

Source: Aleksandar Milenkovic

La très grande famille des ARMs

Les jeux d'instructions - Condition codes

- The possible condition codes are listed below:
  - Note AL is the default and does not need to be specified

CODE	MEANING	FLAGS
EQ	Equal equals zero	Z
NE	Not equal	!Z
VS	Overflow	V
VC	No overflow	!V
MI	Minus/negative	N
PL	Plus/positive or zero	!N
CS	Carry set/unsigned higher or same	C
CC	Carry clear/unsigned lower	!C
HI	Unsigned higher	C and !Z
LS	Unsigned lower or same	!C or Z
GE	Signed greater than or equal	N == V
LT	Signed less than	N != V
GT	Signed greater than	!Z and (N == V)
LE	Signed less than or equal	Z or (N != V)
AL	Always (default)	Any

247

La très grande famille des ARMs

Les jeux d'instructions - Conditional execution

- Conditional execution to avoid branch instructions used to skip a small number of non-branch instructions
- Example

```
        CMP r0, #5      ;
        BEQ BYPASS      ; if (r0!=5) {
        ADD r1, r1, r0  ;    r1:=r1+r0-r2
        SUB r1, r1, r2  ; }
BYPASS: …
```

With conditional execution

```
        CMP r0, #5      ;
        ADDNE r1, r1, r0 ;
        SUBNE r1, r1, r2 ;
        …
```

```
; if ((a==b) && (c==d)) e++;

CMP r0, r1
CMPEQ r2, r3
ADDEQ r4, r4, #1
```

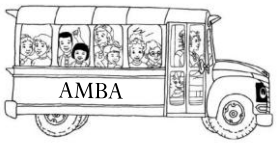
Note: add 2 –letter condition after the 3-letter opcode

248

Source: Aleksandar Milenkovic

Trois petits « extras »

- Organisation “big.LITTLE”
- Un mot sur les bus AMBA



- La nouvelle architecture ARM v8-A

La très grande famille des ARMs

big.LITTLE

- Rappelez-vous (séance 3):

Mono - Multi - Many Cœur(s)

Besoin d'optimiser l'énergie

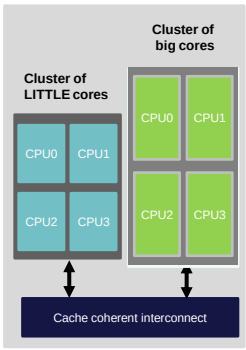
- Comment concilier puissance de calcul et gestion d'énergie?
- La plupart des micros dédiés aux systèmes embarqués sur batterie disposent de mécanismes permettant d'optimiser la consommation en énergie
- Par exemple, modulation de la fréquence de fonctionnement selon les besoins en puissance à un instant t
- Certains micros récents vont jusqu'à embarquer plusieurs types de cœurs.
- Exemple: le tegra X1 de Nvidia (quad core A57 + quad core A53)

C. Brunschweiler

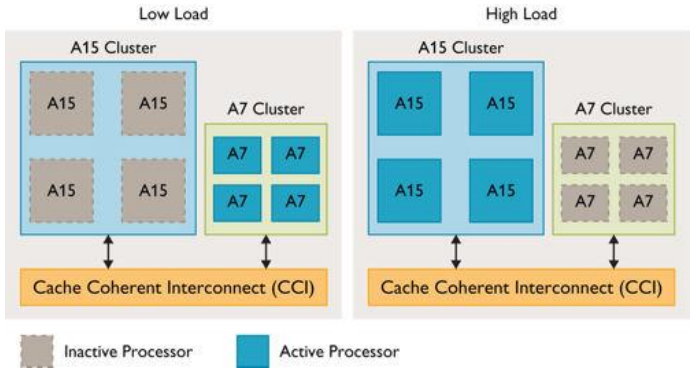
140 Architecture des systèmes embarqués

La très grande famille des ARMs
big.LITTLE

- Principe
  - **two clusters** of cores, as shown below:
    - a cluster of a low power cores, termed as the LITTLE cores and
    - a cluster of high performance cores, termed as the big cores.



La très grande famille des ARMs
big.LITTLE



La très grande famille des ARMs

big.LITTLE

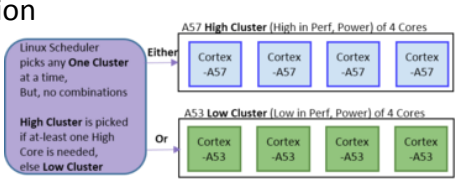
	1st Generation: ARMv7 (32-bit, 40-bit physical)	2rd Generation: ARMv8 (32-bit/64-bit)
High-performance CPU (big)	Cortex-A15, Cortex-A17	Cortex-A57, Cortex-A72
High-efficiency CPU (LITTLE)	Cortex-A7	Cortex-A53

	Cortex-A15 vs Cortex-A7 Performance	Cortex-A7 vs Cortex-A15 Energy Efficiency
Dhrystone	1.9x	3.5x
FDCT	2.3x	3.8x
IMDCT	3.0x	3.0x
MemCopy L1	1.9x	2.3x
MemCopy L2	1.9x	3.4x

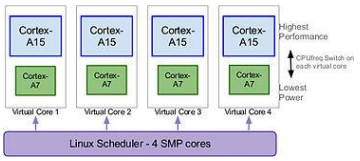
La très grande famille des ARMs

big.LITTLE

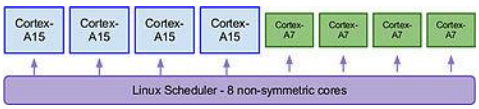
- Clustered migration



- CPU migration

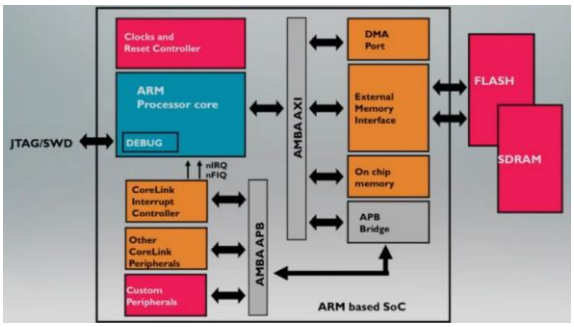


- Heterogeneous multi-processing



La très grande famille des ARMs

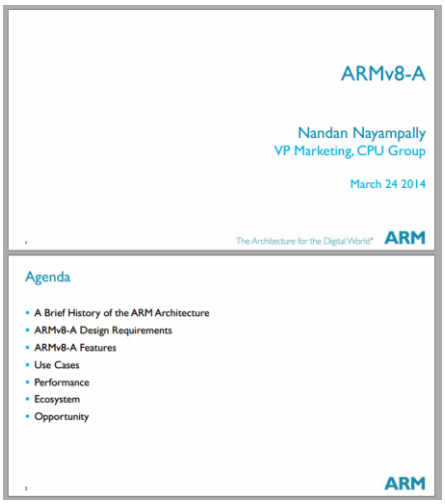
Bus AMBA



- Advanced eXtensible Interface (AXI)
- Advanced Peripheral Bus (APB)
- (Advanced High-performance Bus (AHB))
- (Advanced System Bus (ASB))

La très grande famille des ARMs

La nouvelle architecture ARM V8-A


ARMv8-A

Nandan Nayampally
VP Marketing, CPU Group

March 24 2014

The Architecture for the Digital World® ARM

Agenda

- A Brief History of the ARM Architecture
- ARMv8-A Design Requirements
- ARMv8-A Features
- Use Cases
- Performance
- Ecosystem
- Opportunity

ARM

http://media.corporate-ir.net/media_files/IROL/19/197211/ARMv8-A%20IR%20webcast%2024_03_2014.pdf

Prochaine(s) séance(s)

Mise en œuvre d'un
STM32 à base de Cortex M4

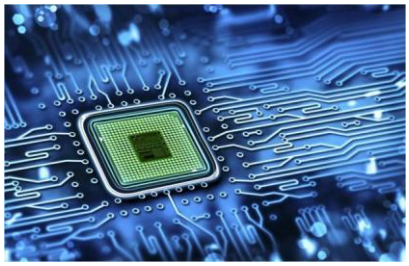
Mise en œuvre d'un STM32 à base de Cortex M4

- Pensez à amener des câbles mini-USB!!

- Et un PC sous Windows…

MERCI ;-)

Plan



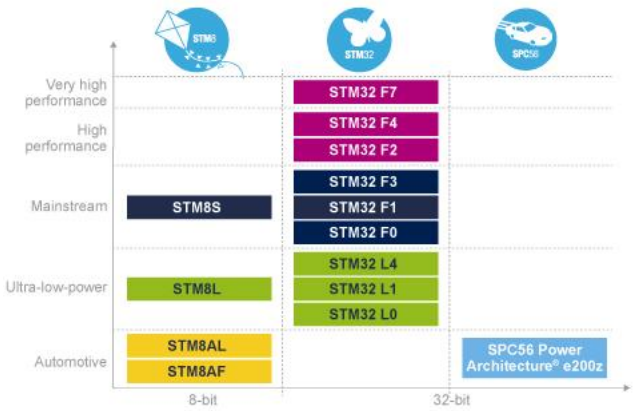
- Introduction
- Un peu d'histoire…
- Rappels théoriques
- Diversité des « puces électroniques »
- Familles de micros
- **Mise en œuvre d'un STM32 à base d'ARM Cortex M4**
- Conception des systèmes embarqués
- Du code au monde physique (par l'étude de quelques cartes – systèmes électroniques)
- Cas concrets: analyse de différents systèmes embarqués

Mise en œuvre d'un STM32 à base d'ARM Cortex M4

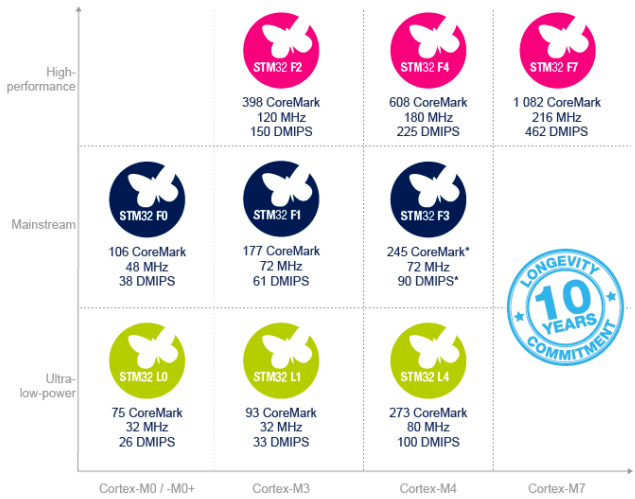
La famille des STM32
Le STM32F401RE et sa carte de développement NUCLEO
La carte d'extension Bluetooth LE BlueNRG
Un mot sur le bus SPI
Place à la pratique

La famille des STM32

- Gamme de microcontrôleurs 32 bits à cœur ARM Cortex M

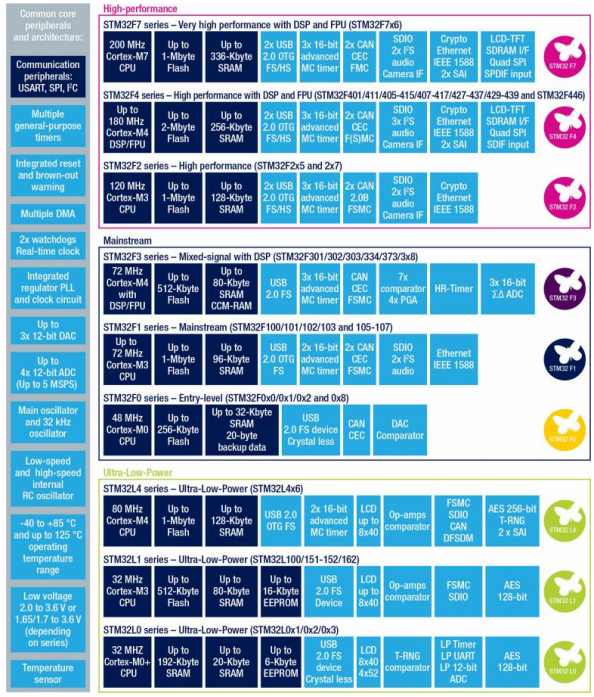


La famille des STM32

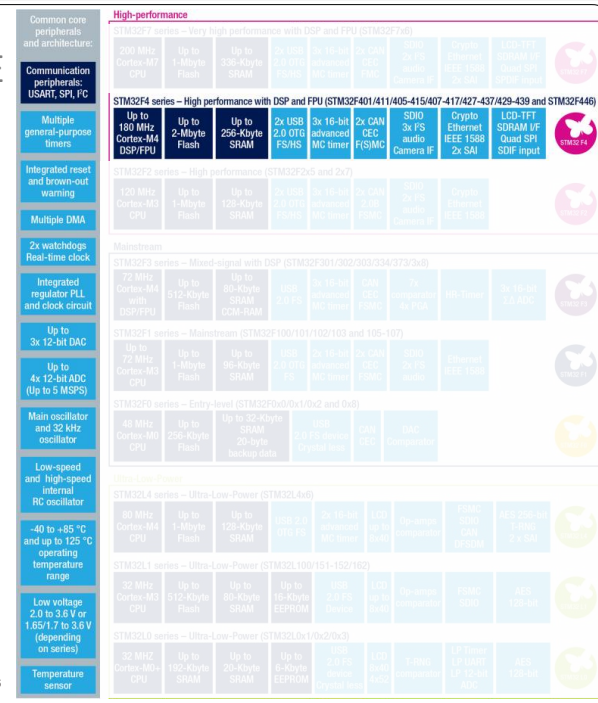


* from CCM-SRAM

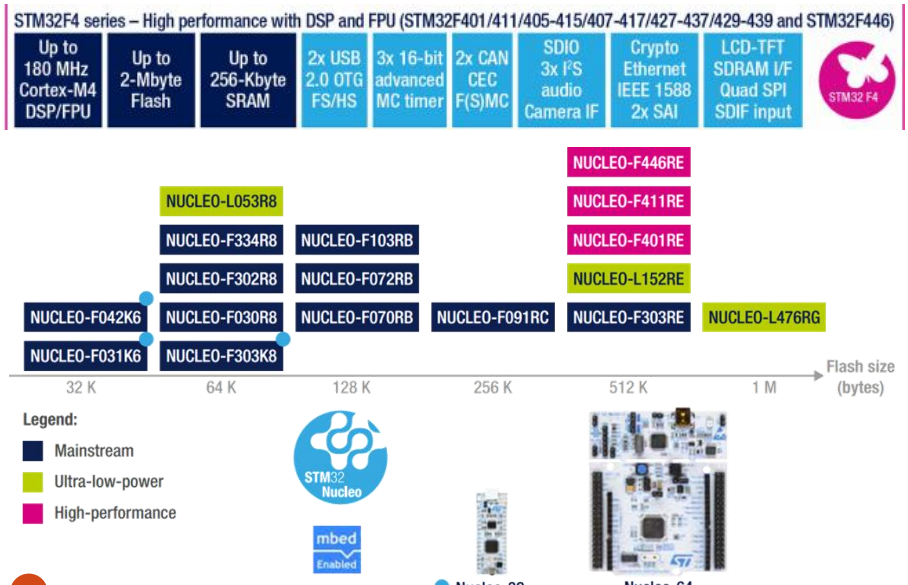
La famille des STM32



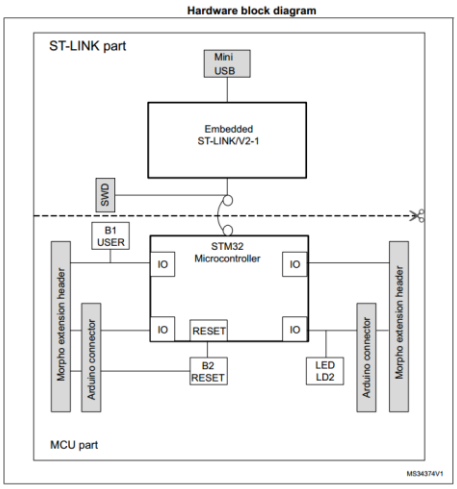
Le STM32F401RE et sa carte de développement NUCLEO



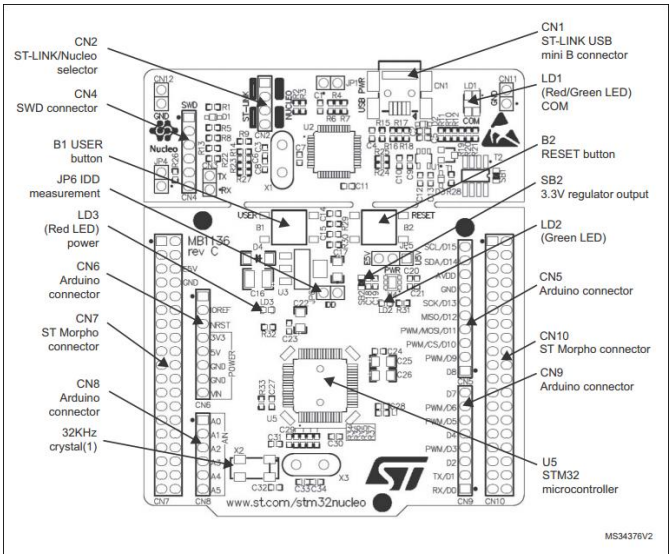
Le STM32F401RE et sa carte de développement NUCLEO



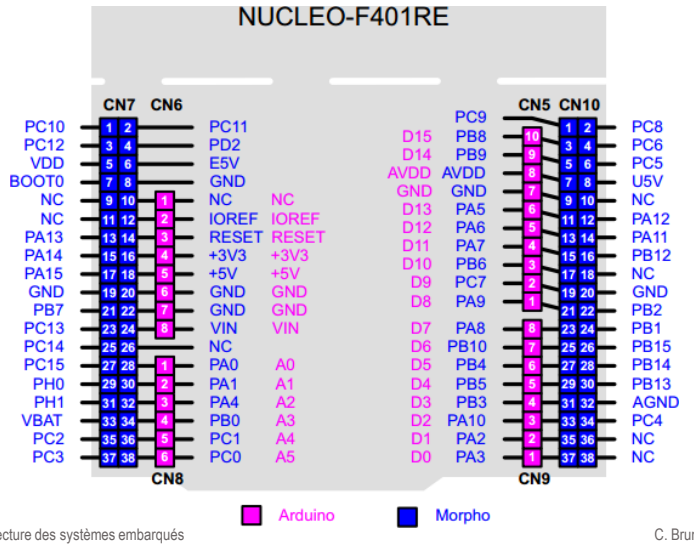
Le STM32F401RE et sa carte de développement NUCLEO



Le STM32F401RE et sa carte de développement NUCLEO



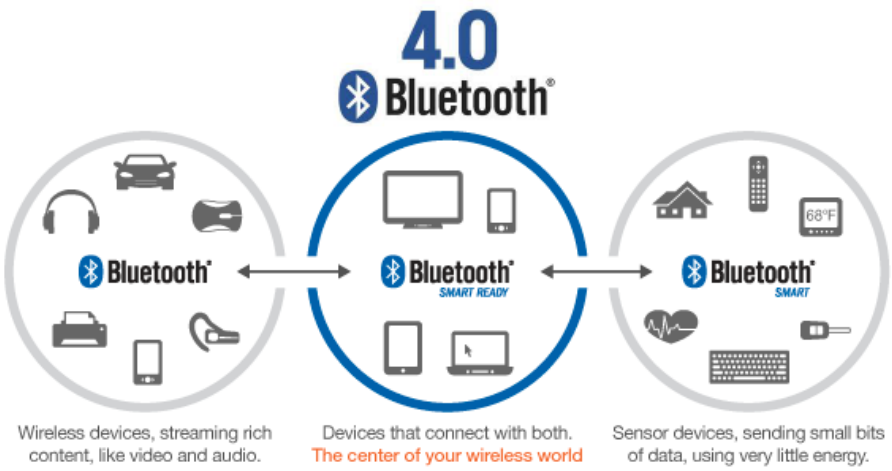
Le STM32F401RE et sa carte de développement NUCLEO



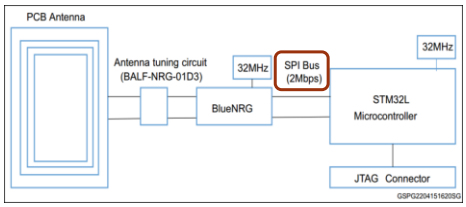
La carte d'extension Bluetooth LE BlueNRG

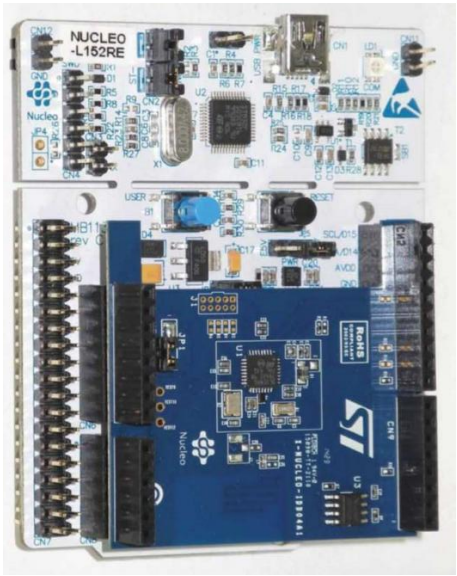
- BlueNRG low power, low energy Bluetooth network coprocessor
- Free comprehensive development firmware library and example for BlueNRG, compatible
- with STM32Cube firmware
- Bluetooth low energy 4.0 master and slave compliant
- Compatible with STM32 Nucleo boards
- Equipped with Arduino UNO R3 connector
- Very low power consumption: 7.3 mA RX and 8.2 mA TX at +0 dBm
- Maximum transmission power: +8 dBm
- Excellent receiver sensitivity (-88 dBm)
- X-NUCLEO-IDB04A1 is FCC certified (FCC ID: S9NIDB04A1)
- X-NUCLEO-IDB04A1 is officially certificated as a BTLE 5 mW module for Japan Radio Law "TYPE" Certification by Japan government
- RoHS compliant

La carte d'extension Bluetooth LE BlueNRG



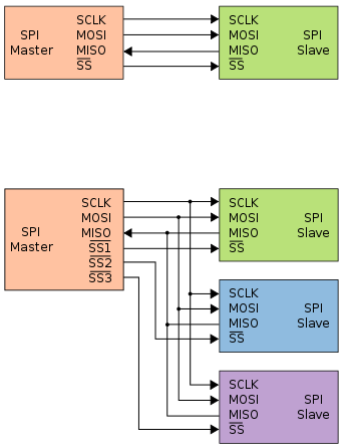
La carte d'extension Bluetooth LE BlueNRG





Un mot sur le bus SPI

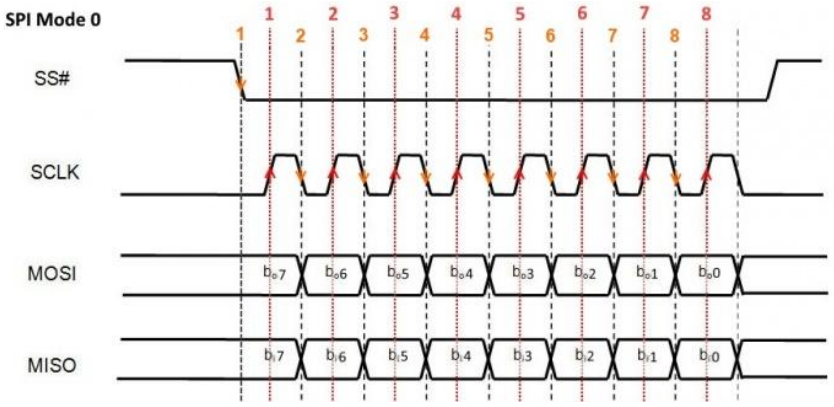
- SPI = Serial Peripheral Interface
- Bus de données série synchrone Full-duplex
- Schéma Maître-Esclave**s**
- Quatre signaux:
  - SCLK - Serial Clock
  - MOSI - Master Out, Slave In
  - MISO - Master In, Slave Out
  - SS - Slave Select



Source: Wikipedia

Un mot sur le bus SPI

- Principe de communication Full-duplex



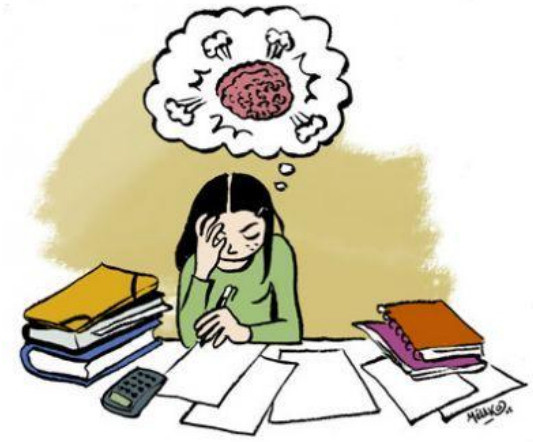
Source: wiki.electroniciens.cnrs.fr

Un mot sur le bus SPI

- Avantages
  - Communication Full duplex
  - Débit assez important par rapport à I2C
  - Flexibilité du nombre de bits à transmettre ainsi que du protocole en lui-même
  - Simplicité de l'interface matérielle
  - Aucun arbitre nécessaire car aucune collision possible
  - Les esclaves utilisent l'horloge du maître et n'ont donc pas besoin d'oscillateur propre
  - Partage d'un bus commun pour l'horloge, MISO et MOSI entre les périphériques
- Inconvénients
  - Monopolise plus de broches d'un boîtier que l'I2C ou une UART qui en utilisent seulement deux.
  - Aucun adressage possible, il faut une ligne de sélection par esclave.
  - Le protocole n'a pas d'acquittement. Le maître peut parler dans le vide sans le savoir.
  - Ne s'utilise que sur de courtes distances contrairement aux liaisons RS-232, RS-485 ou bus CAN.

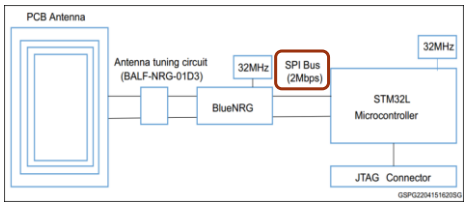
Source: Wikipedia

Exercice



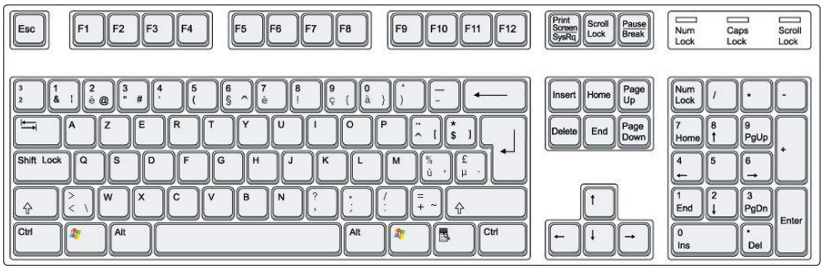
Etude de la liaison SPI entre la carte NUCLEO et la carte BLE

- Grâce à la documentation des cartes NUCLEO et BLE, déterminez quelles interfaces du STM32 sont utilisées pour réaliser la communication avec le composant Blue NRG.



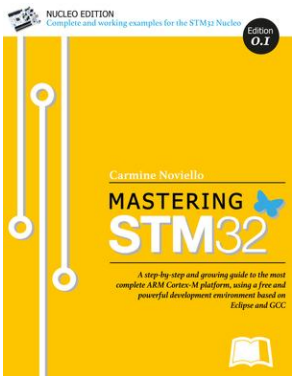
Place à la pratique

- A vos claviers !



Mise en place de l'environnement de développement Eclipse / GCC pour STM32

- Suivre les indications du §2.2 « Installing the tool-chain on Windows » du livre:



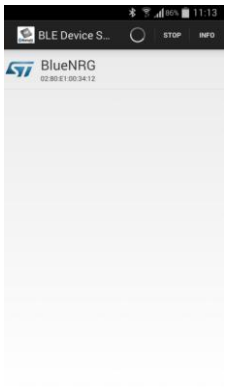
http://samples.leanpub.com/mastering-stm32-sample.pdf

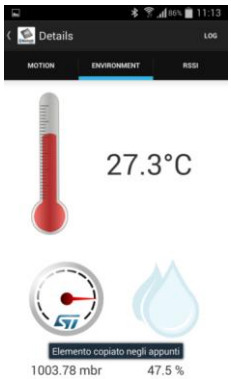
Mettre en œuvre l'exemple d'application Blue NRG

- Suivre les indications présentées sur la page:

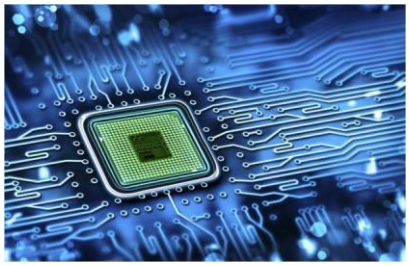
  http://www.carminenoviello.com/en/2015/03/08/bluenrg-shield-stm32-nucleo/

- Pensez à changer le nom de votre périphérique…





Plan

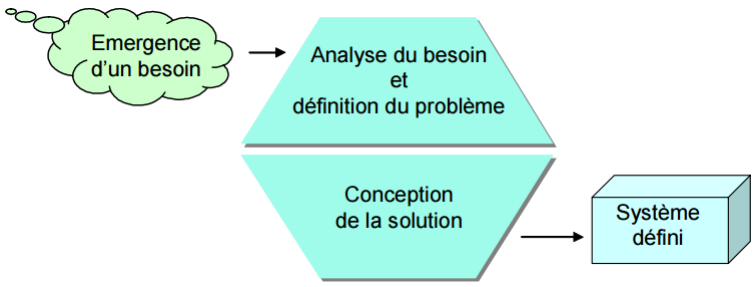


- Introduction
- Un peu d'histoire…
- Rappels théoriques
- Diversité des « puces électroniques »
- Familles de micros
- Mise en œuvre d'un STM32 à base d'ARM Cortex M4
- **Conception des systèmes embarqués**
- Du code au monde physique (par l'étude de quelques cartes – systèmes électroniques)
- Cas concrets: analyse de différents systèmes embarqués

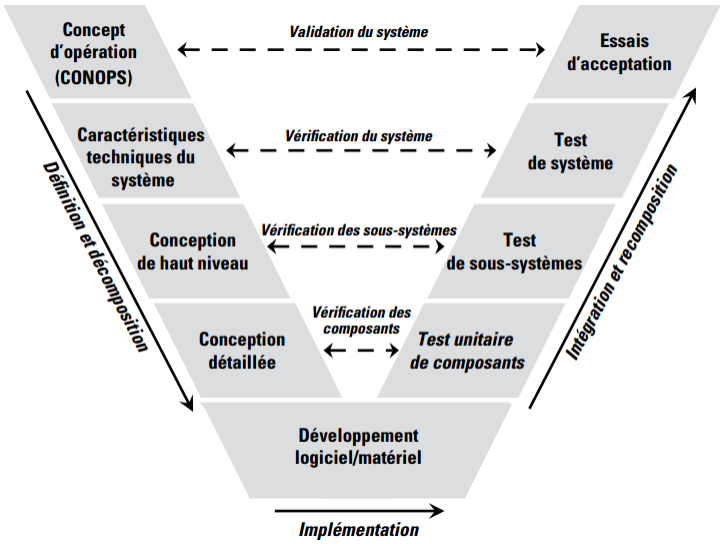
Conception des systèmes embarqués

Des exigences à la validation du système

Conception des systèmes embarqués



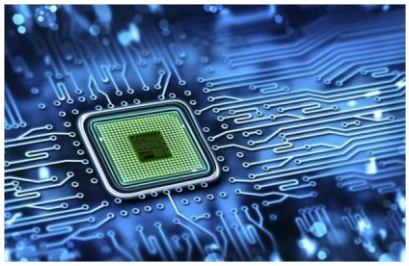
Conception des systèmes embarqués



Conception des systèmes embarqués

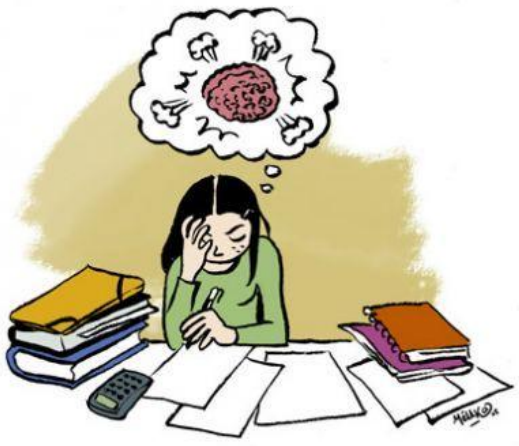
(Thème traité à l'oral)

Plan



- Introduction
- Un peu d'histoire…
- Rappels théoriques
- Diversité des « puces électroniques »
- Familles de micros
- Mise en œuvre d'un STM32 à base d'ARM Cortex M4
- Conception des systèmes embarqués
- **Du code au monde physique (par l'étude de quelques cartes – systèmes électroniques)**
- Cas concrets: analyse de différents systèmes embarqués

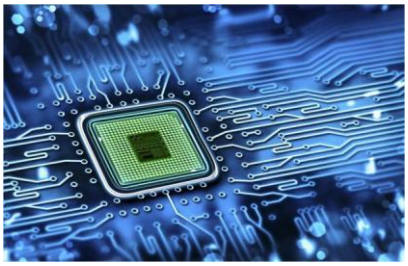
Exercice



Etude de quelques cartes – systèmes électroniques

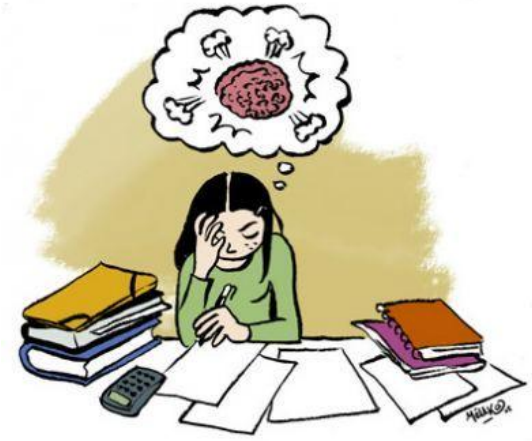
- Grâce au web, dresser un tableau comparatif des solutions de matériel « libre » suivantes:
  - Arduino
  - Raspberry (B, B+, 2)
  - Beaglebone / Beagleboard
  - Udoo
  - Cubieboard / CubieTruck
  - RIoT board
  - PCDuino
  - Hummingboard
  - Odroid
  - Intel Edison
  - …

Plan



- Introduction
- Un peu d'histoire…
- Rappels théoriques
- Diversité des « puces électroniques »
- Familles de micros
- Mise en œuvre d'un STM32 à base d'ARM Cortex M4
- Conception des systèmes embarqués
- Du code au monde physique (par l'étude de quelques cartes – systèmes électroniques)
- **Cas concrets: analyse de différents systèmes embarqués**

Exercice



Analyse de différents systèmes embarqués

- Par groupe de 4 à 5 personnes (4 x 4 + 2 x 5)
- Analyse de 6 systèmes embarqués différents
- Préparation d'une synthèse dans le but de présenter aux autres groupes en fin de séance (durée de présentation: ~ 10 min par système)
- Timing:
  - Analyse et préparation de la synthèse: jusqu'à 16h
  - Présentations: à partir de 16h

Système 1: The Drone Pi

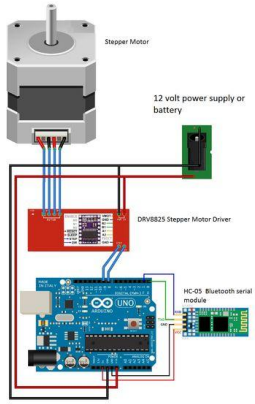
- http://www.instructables.com/id/The-Drone-Pi/

Système 2: l'antivol à tondeuse…

- http://www.domotique-info.fr/2015/03/linkit-one-la-plateforme-iot-de-mediatek/

Système 3: Motorized camera slider controlled by Android phone

- http://www.instructables.com/id/Motorized-camera-slider-controled-by-Android-phone

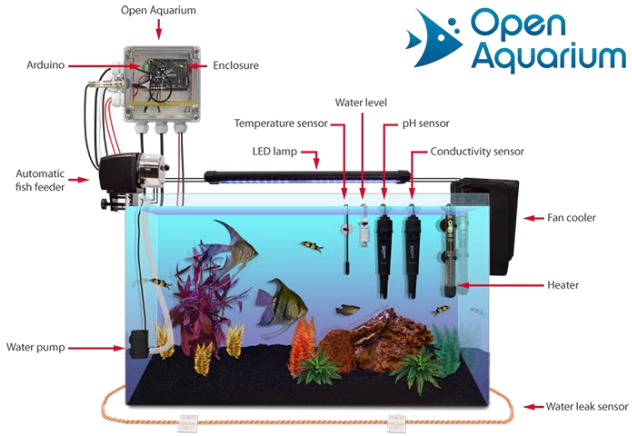


Système 4: Airbeam (…)

- http://aircasting.org/

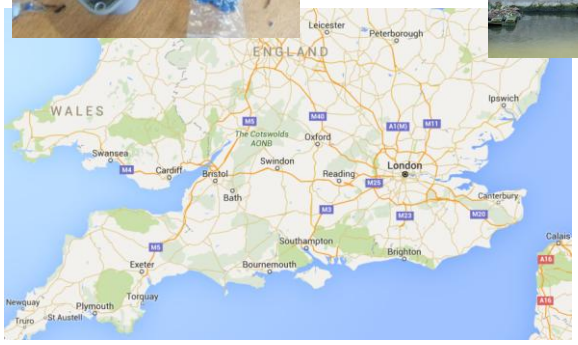
Système 5: Open Aquarium

- https://www.cooking-hacks.com/documentation/tutorials/open-aquarium-aquaponics-fish-tank-monitoring-arduino



Système 6: Oxford Flood Network

- http://oxfloodnet.co.uk/



RDV le 2 décembre pour l'examen ;-)

"The future depends on what you do today."
Mahatma Gandhi