Introduction à l'architecture ARM

GIF-1001 Ordinateurs: Structure et Applications

Jean-François Lalonde

Petit historique

- Tandis qu'Intel produit des microprocesseurs 8086 et 80286, des employés d'une compagnie anglaise inspirés par les travaux de l'heure travaillent à la conception d'un ordinateur

- En 1985, la compagnie ACORN produit un premier processeur qui résulte de ces travaux: le Acorn RISC Machine (ARM)

- La complexité du ARM est réduite par choix et par faute de moyens et le jeu d'instructions est simplifié.

- Alors que le 8086 gagne en popularité, ACORN et Apple collaborent pour fonder la compagnie ARM ltd en 1990

- ARM et Intel feront évoluer leurs produits en parallèle dans des marchés différents

ARM

- ARM (Advanced RISC Machine) Holdings:
  - développe des architectures de micro-processeurs et des jeux d'instructions
  - ne construit aucun micro-processeur comme tel! La compagnie licencie la technologie à d'autres qui les fabriquent à leur façon
    - clients: Apple, Nvidia, Samsung, Texas Instruments, etc…
- Micro-processeurs ARM
  - Supportent 32 et 64 bits
  - L'architecture **la** plus utilisée au monde
    - 100 milliards de processeurs produits en 2017

Processeurs ARM

- Plusieurs versions de processeurs, utilisées partout!
  - ARM7TDMI(-S): Nintendo DS, Lego NXT
  - ARM946E-S: Canon 5D Mark ii (caméra)
  - ARM1176JZ(F)-S: Raspberry Pi
  - Cortex-A9: Apple iPhone 4S, iPad2
  - Cortex-A15: Nexus 10
- Beaucoup d'autres!
  - http://en.wikipedia.org/wiki/List_of_applications_of_ARM_cores



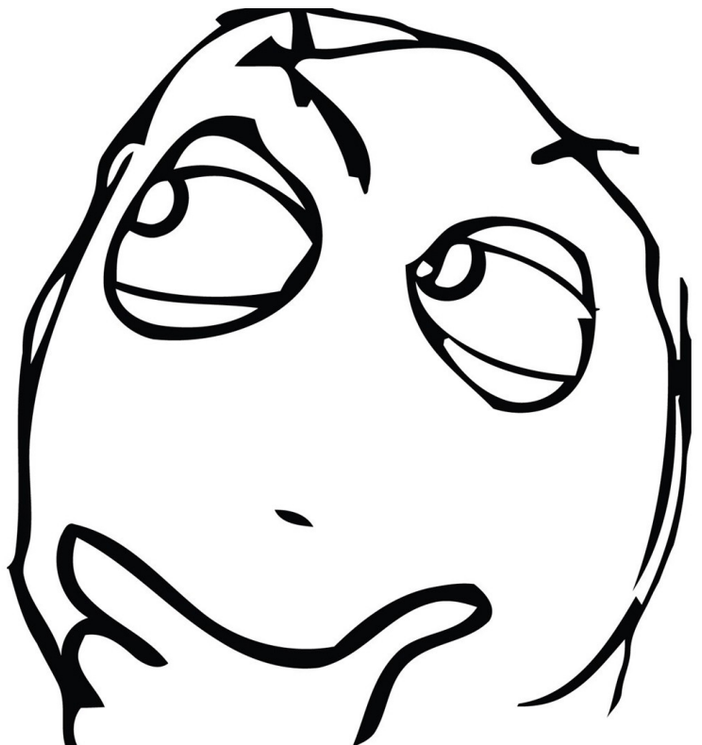


Architecture des ARM (32 bits)

- Dans le cours: ARM7TDMI (architecture 32 bits populaire)
  - Registres de 32 bits
  - Instructions de 32 bits
  - Adresses de 32 bits
- Architecture RISC pour laquelle tout passe par des registres

3 Affaires à Retenir Mentalement™ (ARM)

- ARM™ #1 : Organisation de la mémoire
- ARM™ #2 : Instructions et PC
- ARM™ #3 : Registres



ARM™ #1: Organisation de la mémoire

	TP1	ARM
Taille d'un emplacement mémoire	2 octets (16 bits)	1 octet (8 bits)
Largeur du bus de données	2 octets (16 bits)	4 octets (32 bits)
Pour stocker un mot, on a besoin de:	1 adresse	4 adresses

Question

Comment stocker un mot de 4 octets à l'adresse 0x0, sachant que chaque adresse stocke 1 octet?

exemple: 0x12345678?

Adresse	Octet
0x0	
0x1	
0x2	
0x3	
0x4	
…	

Adresse	Octet
0x0	
0x1	
0x2	
0x3	
0x4	
…	

Question

Comment stocker un mot de 4 octets à l'adresse 0x0, sachant que chaque adresse stocke 1 octet?

exemple: 0x12345678?

Adresse	Octet
0x0	12
0x1	34
0x2	56
0x3	78
0x4	…
…	…

Adresse	Octet
0x0	78
0x1	56
0x2	34
0x3	12
0x4	…
…	…

Mémoire

Adresse	Octet
0x0	12
0x1	34
0x2	56
0x3	78
0x4	…
…	…

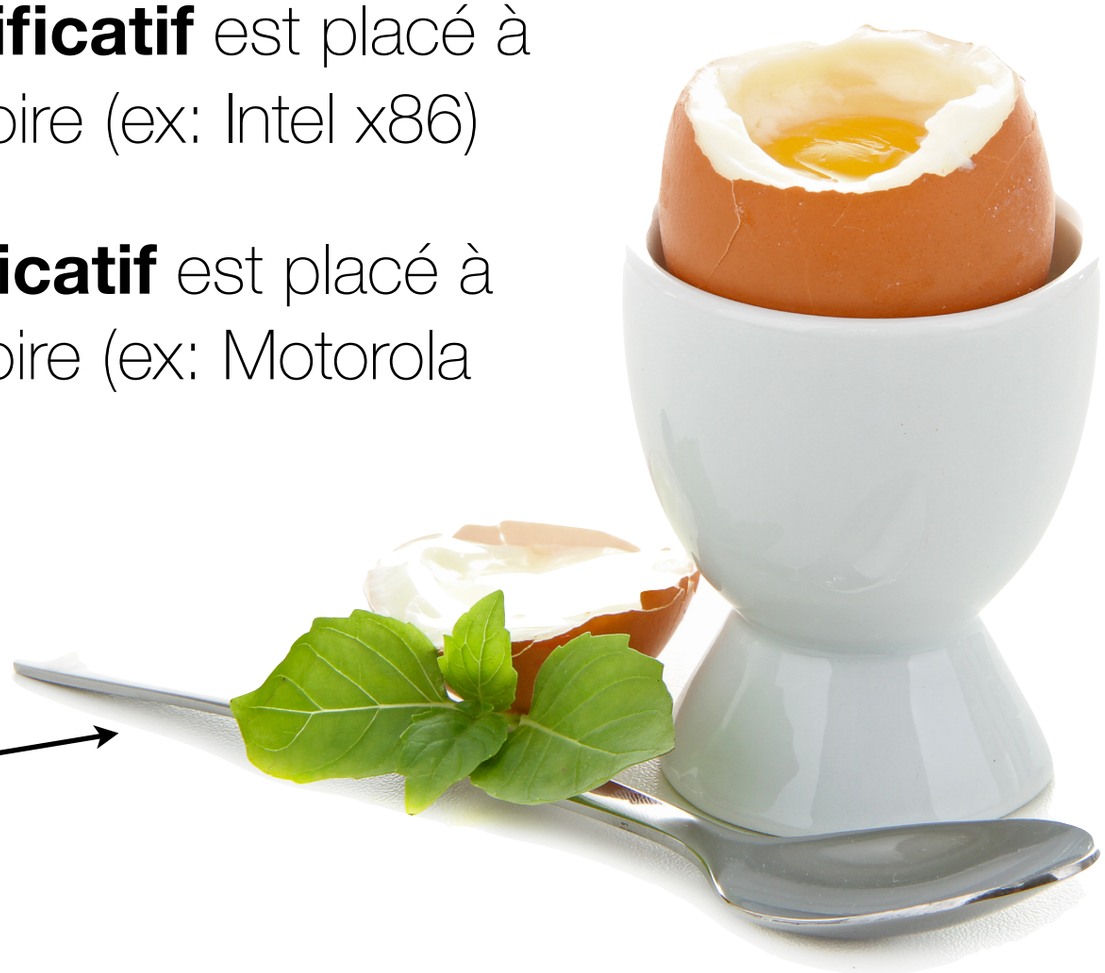
Adresse	+0	+1	+2	+3
0x00	12	34	56	78
0x04	…	…		
0x08				
0x0C				
0x10				

Big vs *Little Endian*

- La taille minimum d'une donnée stockée en mémoire est habituellement 1 octet

- Les données ayant plusieurs octets peuvent être stockées de 2 façons:
  - *Little endian* : **l'octet le moins significatif** est placé à la **plus petite** adresse dans la mémoire (ex: Intel x86)
  - *Big endian* : **l'octet le moins significatif** est placé à la **plus haute** adresse dans la mémoire (ex: Motorola 68000)

Pourquoi l'oeuf?
Voyez « Les Voyages de Gulliver » par Jonathan Swift, où deux tribus de lilliputiens s'affrontent afin de déterminer laquelle des deux mange ses oeufs à la coque de la « bonne » façon… le bout le plus large en haut ou en bas? C'est de là que le terme « big/little endian » proviendrait.



Big vs *Little Endian*

Comment stocker un mot de 32 bits (4 octets) à l'adresse 0x0?
exemple: 0x12345**678**?

Big Endian

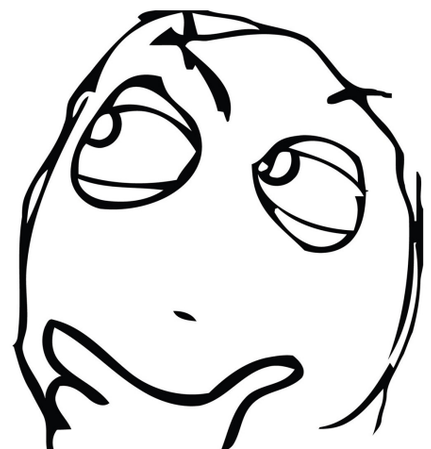
Adresse	**+0**	**+1**	**+2**	**+3**
0x00	12	34	56	**78**
0x04	…	…		

Little Endian

Adresse	**+0**	**+1**	**+2**	**+3**
0x00	**78**	56	34	12
0x04	…	…		

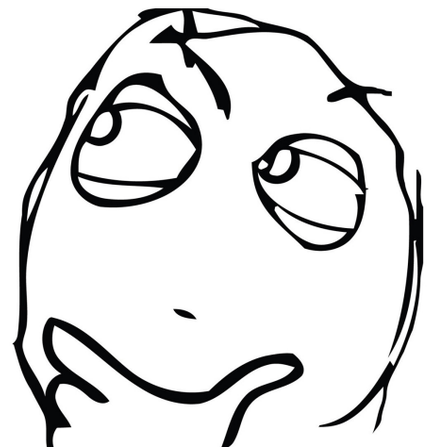
ARM™ #1: Organisation de la mémoire

- Chaque **octet (8 bits)** possède une adresse
- Taille du bus de données, des instructions et des registres: **4 octets (32 bits)**
- Stocke les données en *Little Endian*.



ARM™ #2: Instructions et PC

- Combien d'adresses mémoire a-t-on besoin pour stocker une instruction?
  - chaque octet possède une adresse
  - une instruction possède 4 octets (32 bits)
  - donc **1** instruction nécessite **4** adresses mémoire



PC #1 : incrémentation

Après chaque exécution, de combien doit-on incrémenter PC?

Adresse	Instruction
0x00	MOV R0, #0x40
0x04	LDR R0, [R0]
0x08	ADD R0, R1, R2
0x0C	…
0x10	

PC = PC + 4

PC #2: adresse

Qu’est-ce que contient PC?

L’adresse de la prochaine instruction à être **lue**.

Pipelines

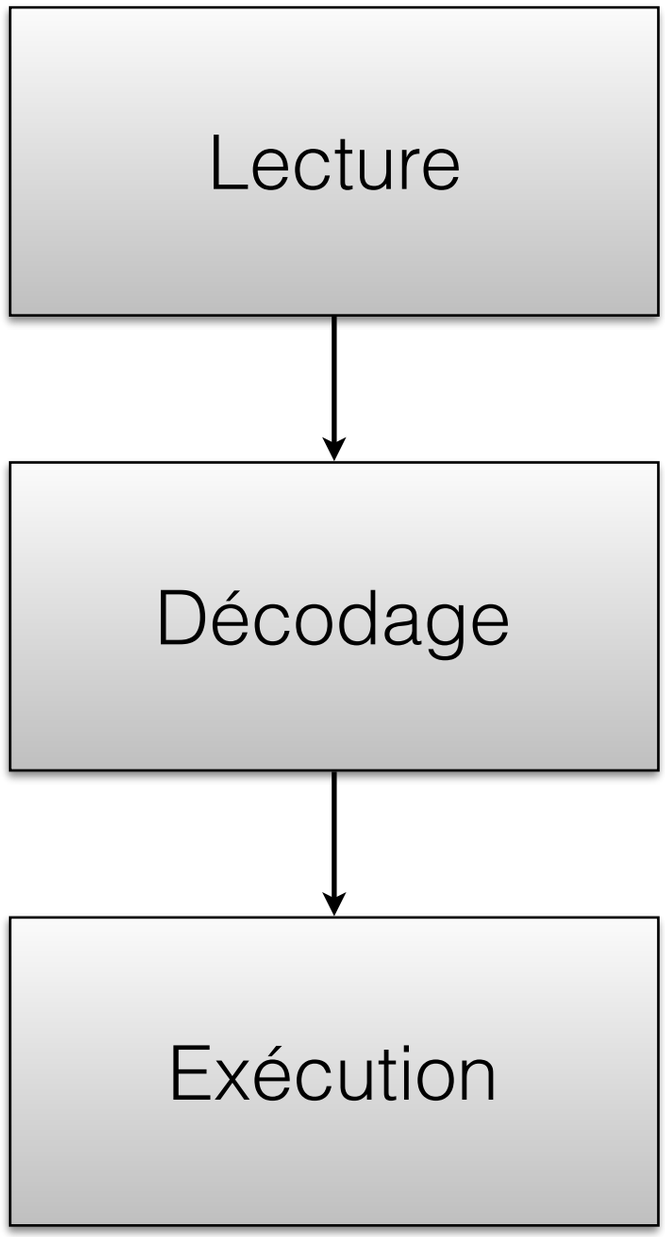
- Début des années 1990
- Idée: séparer l'architecture en plusieurs unités séparées, pouvant être exécutées simultanément
- Comme dans un restaurant (2 unités):
  - *Fetch-decode* : les serveurs vont prendre les commandes des clients aux tables
  - *Execute* : les cuisiniers préparent les repas

Pipelines

Source

Pipeline ARM

- Le pipeline ARM est divisé en 3
- Que contient PC?
  - L'adresse de la prochaine instruction **à être lue**
- Donc, lors de l'exécution d'une instruction, PC indique l'adresse de **deux instructions** plus loin!

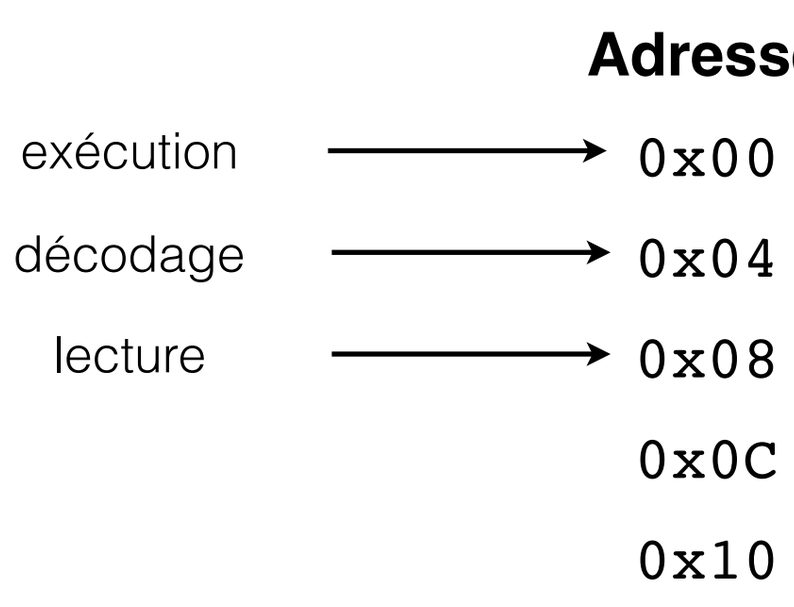


Pipeline ARM

Adresse	Instruction
0x00	MOV R0, #0x40
0x04	LDR R0, [R0]
0x08	ADD R0, R1, R2
0x0C	…
0x10	

Pipeline ARM

- Le pipeline ARM est divisé en 3
- PC contient l'adresse de la prochaine instruction **à être lue**
- Donc, lors de l'exécution d'une instruction, PC indique l'adresse de **deux instructions** plus loin!

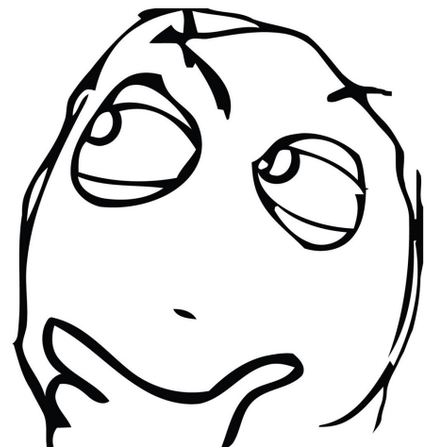


Instruction
`MOV R0, #0x40`
`LDR R0, [R0]`
`ADD R0, R1, R2`
…

PC contient l'adresse de l'instruction courante (exécutée) **+ 8!**

ARM™ #2: Instructions et PC

- Lorsque PC est incrémenté (pour passer à l'instruction suivante), il est incrémenté de **4**
  - PC = **PC + 4**
- PC indique l'adresse de la prochaine instruction à être **lue**
  - PC contient l'adresse de l'instruction courante (exécutée) **+ 8**



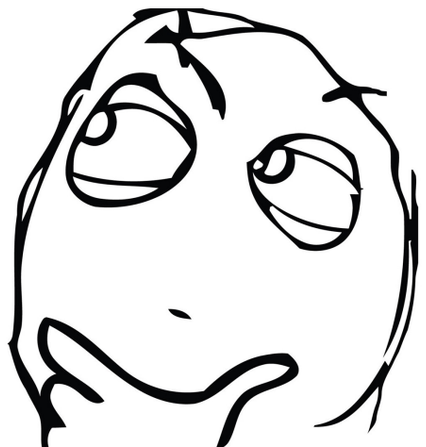
ARM™ #3: Registres

- 16 registres de 32 bits:
  - R0 à R12: registres généraux
  - R13 à R15: registres spécifiques (ayant des fonctionnalités pré-établies)
    - R13: Pointeur de pile (*Stack Pointer* ou SP)
    - R14: Registre de liens (*Link Register* ou LR)
    - R15: Compteur de programme (*Program Counter* ou PC)
- 1 registre d'état, le *Current Program Status Register* (CPSR)
  - mémorise les résultats d'opérations arithmétiques
  - stocke le « mode d'exécution » (on y reviendra bientôt!)

Note

Il est permis de se servir des registres spécifiques comme des registres généraux. Il faut cependant **faire très attention**, car cela peut affecter le comportement du programme de façon indésirée!

Nous vous recommandons donc **d'utiliser R0 à R12** pour vos calculs.



3 Affaires à Retenir Mentalement™ (ARM)

1. Organisation de la mémoire
    - Chaque **octet** possède une adresse, mais les mots possèdent **4 octets**.
    - Mots stockés en mémoire en *little endian*.
2. Instructions et PC
    - PC = PC **+ 4**
    - PC indique l'adresse de l'instruction courante **+ 8**
3. Registres
    - 16 registres
        - 13 registres généraux (R0 à R12)
        - 3 registres spécifiques: SP, LR, PC
    - 1 registre d'état (CPSR)