Introduction à l'architecture ARM

GIF-1001 Ordinateurs: Structure et Applications, Hiver 2016

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

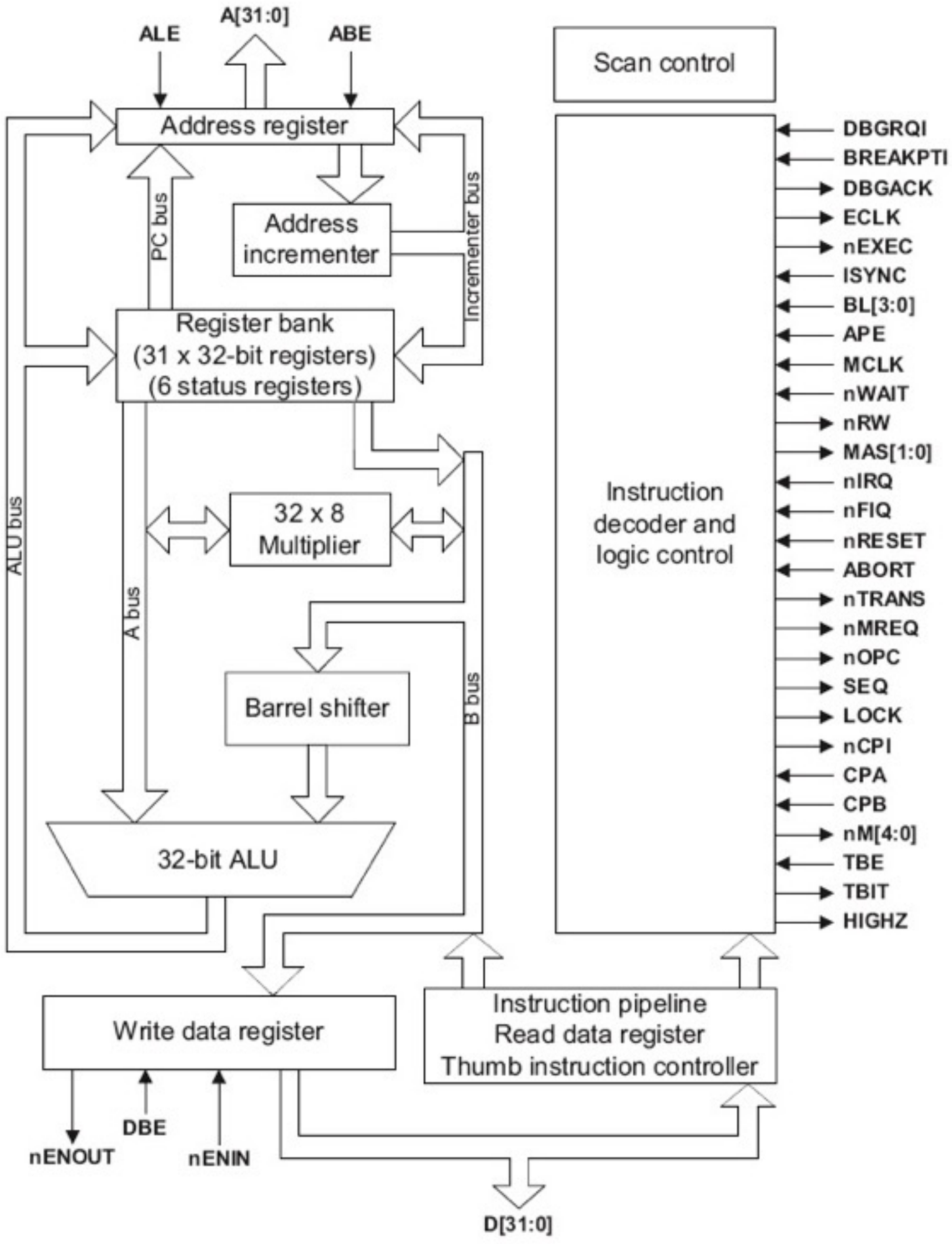
- ARM Holdings:
  - développe des architectures de micro-processeurs et des jeux d'instructions
  - ne construit aucun micro-processeur comme tel! La compagnie licencie la technologie à d'autres qui l'implémentent à leur façon en hardware.
    - clients: Apple, Nvidia, Samsung, Texas Instruments, etc…
- Micro-processeurs ARM
  - Supportent 32 et 64 bits
  - L'architecture **la** plus utilisée au monde
    - 10 milliards produits en 2013
    - 98% de téléphones portables contiennent au moins 1 processeur ARM

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

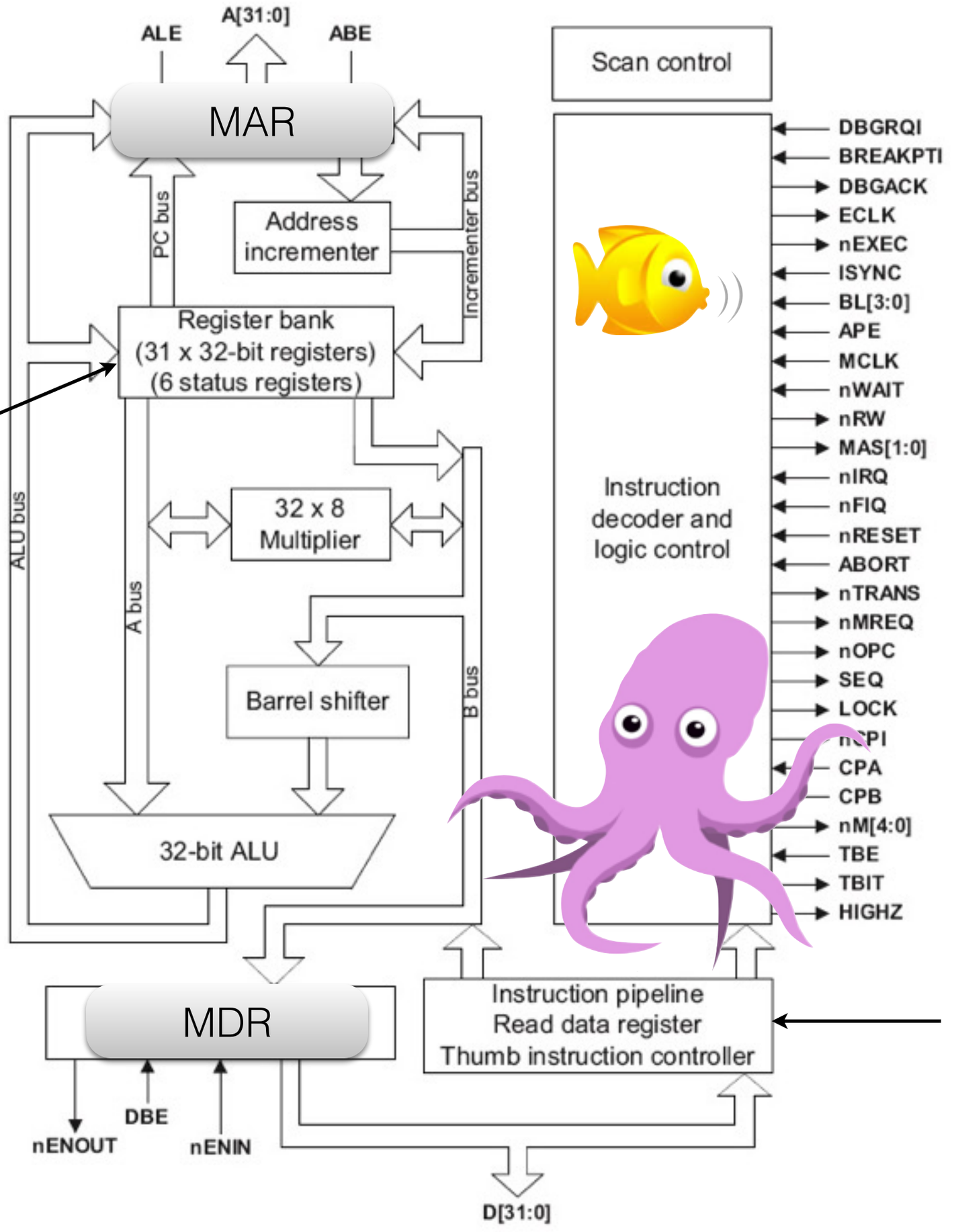
Architecture des ARM (32 bits)

- 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
- Accès à combien de mémoire?
  - 2^{32} octets (4GB) de mémoire



http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0210c/Cacbbajb.html

16 accessibles
« à la fois »



pipeline?

Dans la série «Vous n’avez rien à faire ce weekend?»

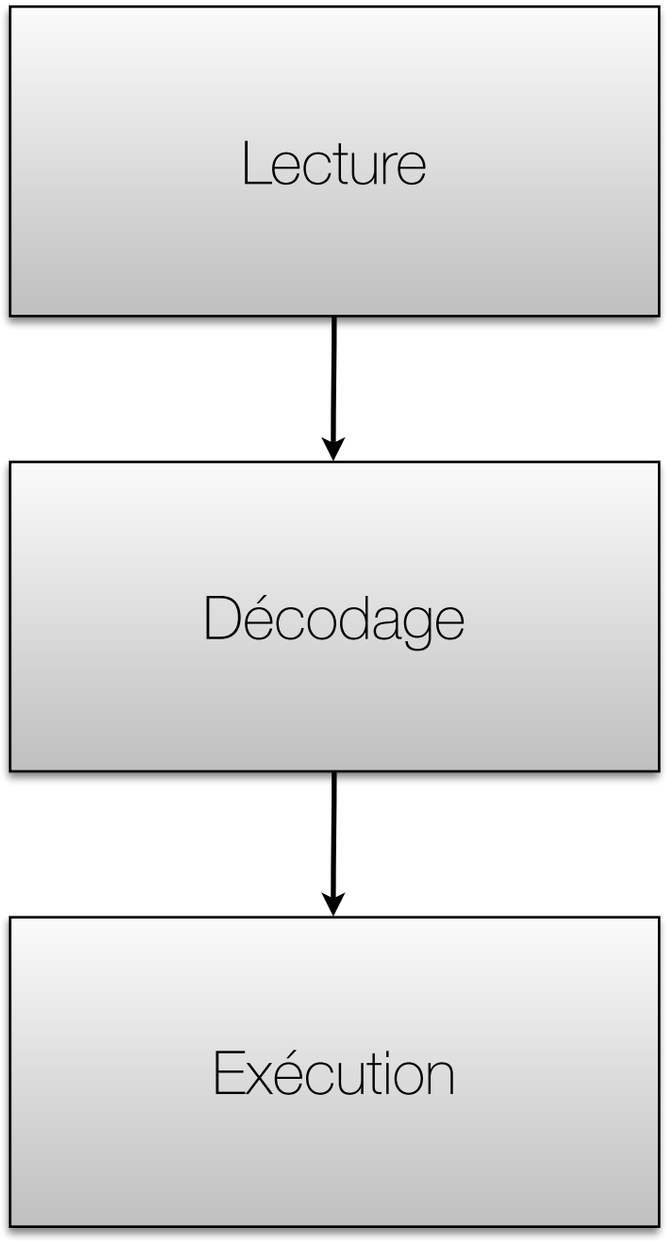
http://www.megaprocessor.com

Pipelines

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

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!



Organisation de la mémoire

TP1

- Taille des mots: **16** bits
- Chaque mot possède une adresse
- Taille du bus de données et des registres: **16** bits
- Taille d'une instruction: **16** bits

ARM

- Taille des mots: **8** bits
- Chaque mot possède une adresse
- Taille du bus de données et des registres: **32** bits
- Taille d'une instruction: **32** bits

Mémoire, instructions, 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 une instruction nécessite 4 adresses mémoire

- 2 considérations importantes sur PC:
  - après chaque exécution, de combien doit-on incrémenter PC?
    - PC = PC **+ 4**
  - qu'est-ce que contient PC?
    - PC = adresse de deux instructions plus loin que l'instruction exécutée
    - PC = adresse de l'instruction exécutée **+ 8**

Pipeline ARM

exécution →
décodage →
lecture →

Adresse	Instruction
0x0	MOV R0, #0x40
0x4	LDR R0, [R0]
0x8	ADD R0, R1, R2
0xC	…

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

Big vs Little Endian

comment stocker un mot de 32 bits (4 octets) à l'adresse 0x0?
exemple: 0x12345678?

Adresse (A)	A+0	A+1	A+2	A+3
0x0	78	56	34	12
0x4				

ou

Adresse (A)	A+0	A+1	A+2	A+3
0x0	12	34	56	78
0x4				

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)

Big vs Little Endian

comment stocker un mot de 32 bits (4 octets) à l'adresse 0x0?
exemple: 0x12345678?

Little Endian

Adresse (A)	A+0	A+1	A+2	A+3
0x0	78	56	34	12
0x4				

Big Endian

Adresse (A)	A+0	A+1	A+2	A+3
0x0	12	34	56	78
0x4				

Les registres

- 16 registres de 32 bits sont disponibles:
  - R0 à R12: usage général
  - R13 à R15: registres “spéciaux”
- Le “Current Program Status Register” (CPSR) est utilisé pour mémoriser les résultats d’opération

Le registres spéciaux R13 à R15

- R13: Pointeur de pile (Stack Pointer ou SP)
- R14: Registre de liens (Link Register ou LR)
- R15: Compteur de programme (Program Counter ou PC)

Le registre “Program Counter” R15

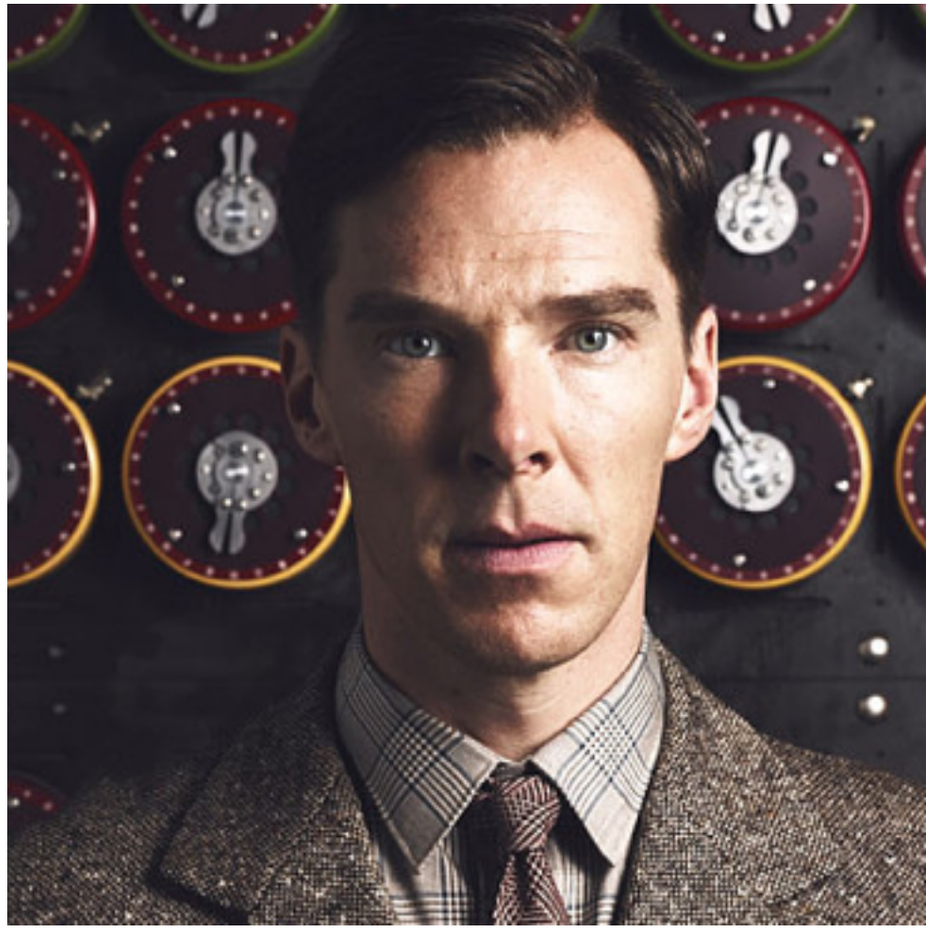
- Le contenu du PC indique à quel endroit de la mémoire se trouve la prochaine exécution que le processeur doit *lire*
- PC identifie des octets tandis que la mémoire est composée de mots de 4 octets
- Il y a 2^{32} octets qui peuvent être adressés donc $2^{32}/4 = 2^{30}$ mots de 4 octets à représenter avec le registre PC
    - C’est dire que les deux bits de poids le moins significatif du registre PC ne servent pas
- 2 considérations importantes:
    - après chaque exécution, de combien doit-on incrémenter PC?
        - PC = PC **+ 4**
    - qu’est-ce que contient PC?
        - PC = adresse de deux instructions plus loin que l’instruction exécutée
        - PC = adresse de l’instruction exécutée **+ 8**

Piles (“stacks”)

- Structure de données très utilisée en informatique
- Inventée par Alan Turing (1946)

Alan Turing

Imitation Game



Piles (“stacks”)

- Structure de données “LIFO”
  - LIFO = Last In, First Out
- Deux opérations principales:
  - PUSH: rajoute un élément sur “le dessus” la pile
  - POP: enlève un élément du “dessus” de la pile

Piles (“stacks”)

- Exemples?
  - Notation Polonaise Inverse dans les calculatrices (“RPN”)
  - Algorithme pour naviguer un labyrinthe
- Stack overflow!

Le “Stack Pointer” (R13)

- Par convention, on définit un bloc de mémoire qu’on nomme “pile” (stack)
- La pile est utilisée pour sauvegarder temporairement des valeurs de travail
  - exemples: variables locales, paramètres et valeurs de retour des fonctions
- Le Stack Pointer contient une adresse de la pile à laquelle on peut écrire ou lire une valeur
- Changer la valeur de SP directement peut être dangereux! On utilise plutôt des instructions dédiées:
  - PUSH: rajouter un élément sur la pile
  - POP: enlever un élément de la pile

Appel de fonction simple (sans paramètres)

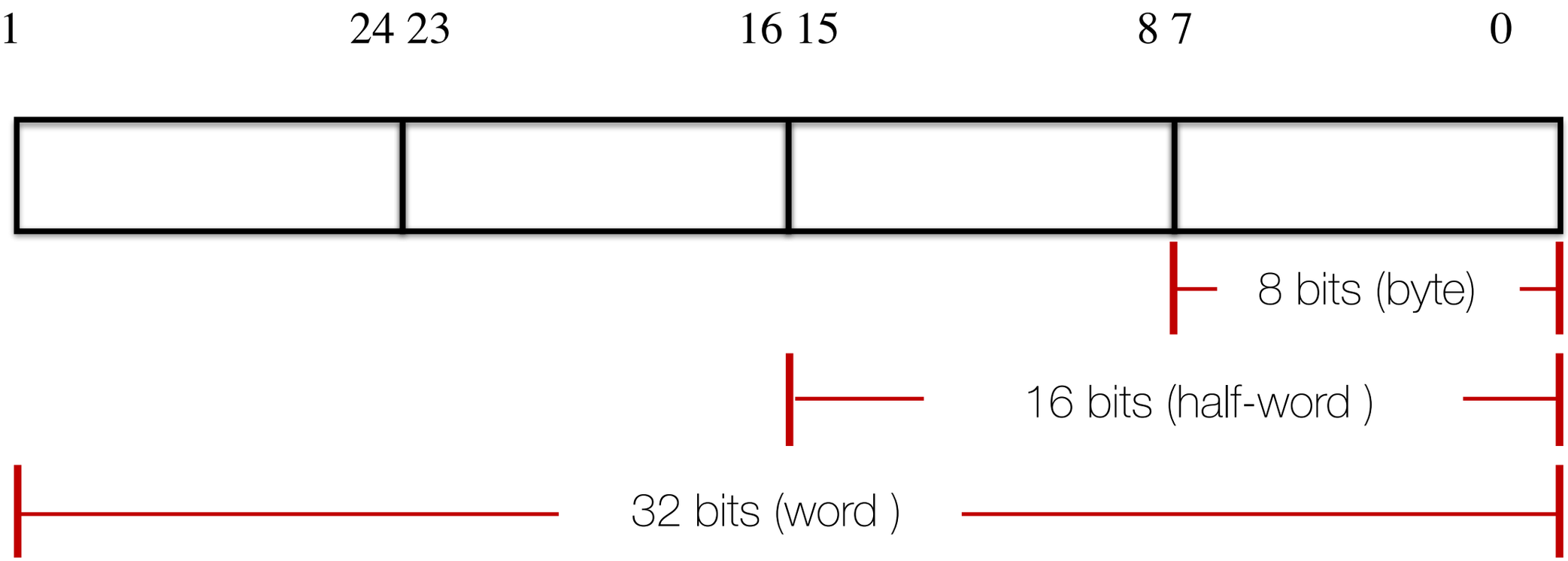
- Mnémonique assembleur indique un appel de fonction (ex: B maFonction)
  - maFonction:
    - INST 1 ; première instruction
    - … ; liste d'instructions
    - … ; "return", indique que la fonction est terminée
- Il suffirait de modifier le PC pour le mettre à l'adresse correspondant aux instructions de "maFonction"
- Cependant, que faire lorsque la fonction est terminée? Où doit-on revenir?

Le Link Register (R14)

- Un appel de fonction à l'aide d'instructions dédiées entraîne un changement de la valeur du Program Counter
- Un retour de fonction ne peut se faire que si PC reprend sa valeur d'avant l'appel de la fonction
- Par convention, le Link Register est utilisé pour sauvegarder l'adresse de retour pendant l'exécution de la fonction
- Mettre la valeur de LR dans PC commande un retour de fonction
- Utiliser LR à d'autres fins ne peut que causer des problèmes si on n'a pas le plein contrôle du programme
- L'appel d'une fonction par une fonction commande une gestion de LR qui se fait avec la pile par convention

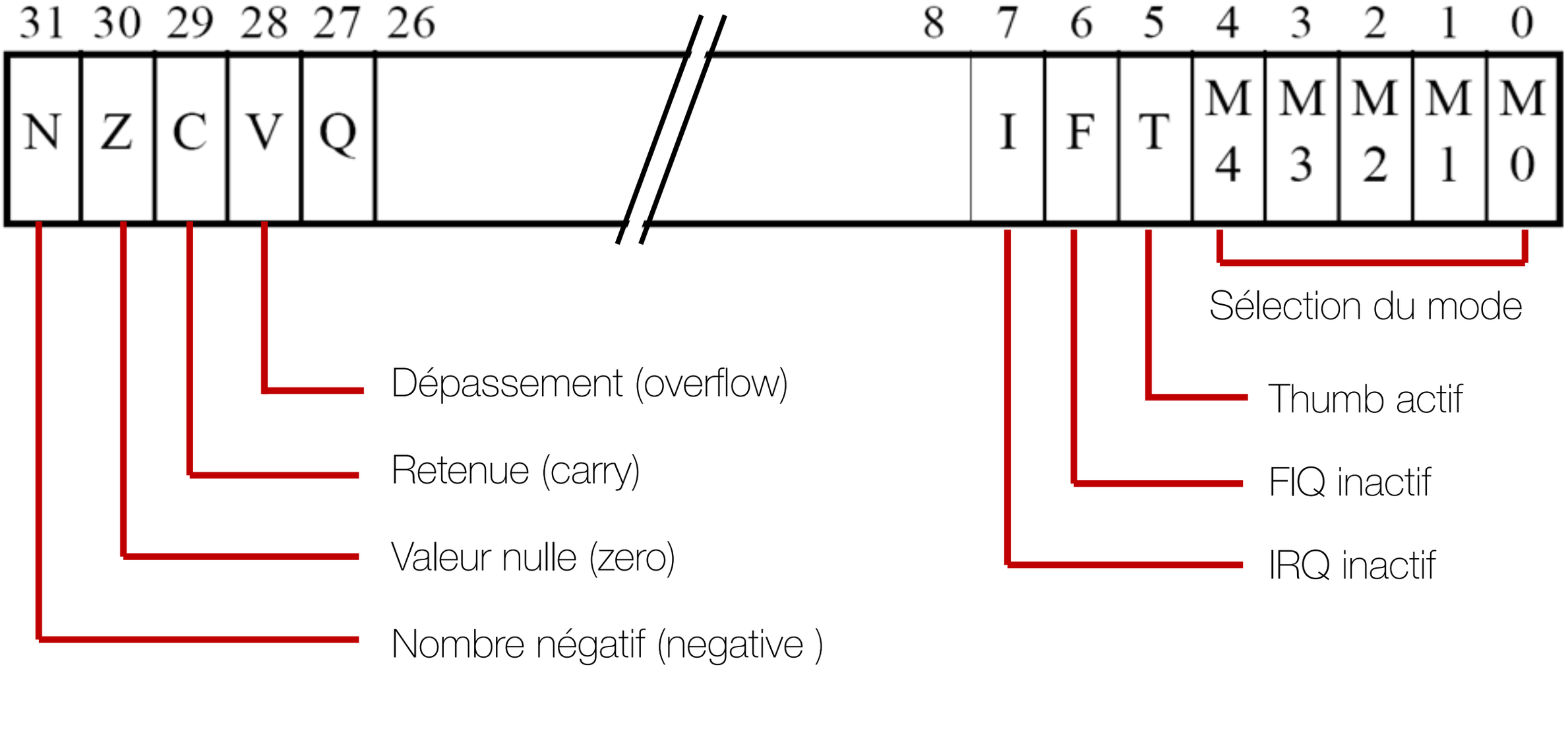
Registres d'usage général

- Gestion des nombres signés ou non à 8, 16 ou 32 bits
- Les calculs sont faits à 32 bits et la gestion des valeurs plus petites est logicielle



Registre de statut (CPSR)

- Un registre de statut décrit l'état du processeur



CPSR: Détection de conditions

- N: Détection de signe négatif
  - 1 si résultat < 0, 0 autrement
- Z: Détection de zéro
  - 1 si résultat = 0, 0 autrement
  - Souvent utilisé pour détecter les égalités
- C: Détection de retenue (“carry)”
  - 1 si l’opération a impliqué une retenue
  - Ex. retenue d’addition de nombres positifs
- V: Détection de dépassements (overflow)
  - 1 si l’opération a impliqué un dépassement
  - Ex. dépassement signé lors d’une addition

Condition spéciale: retenue (« carry »)

- Addition: lorsqu'il y a une retenue
- Exemple 1: 10 + 8 = ? avec nombres non-signés sur 4 bits:
  - $1010_b + 1000_b = 1\ 0010_b$
    Nous avons besoin d'un 5e bit pour que le résultat soit valide, c'est le bit "carry"!
- Exemple 2: -2 + 2 = ? avec nombres signés sur 4 bits (complément-2):
  - 1110b + 0010b = $1\ 0000_b$
    Le résultat est valide sur 4 bits, mais un 5e bit est activé: il y a « carry »!
  - Est-ce qu'il y a débordement?
    - Le bit de signe n'est pas inversé, pas de débordement!