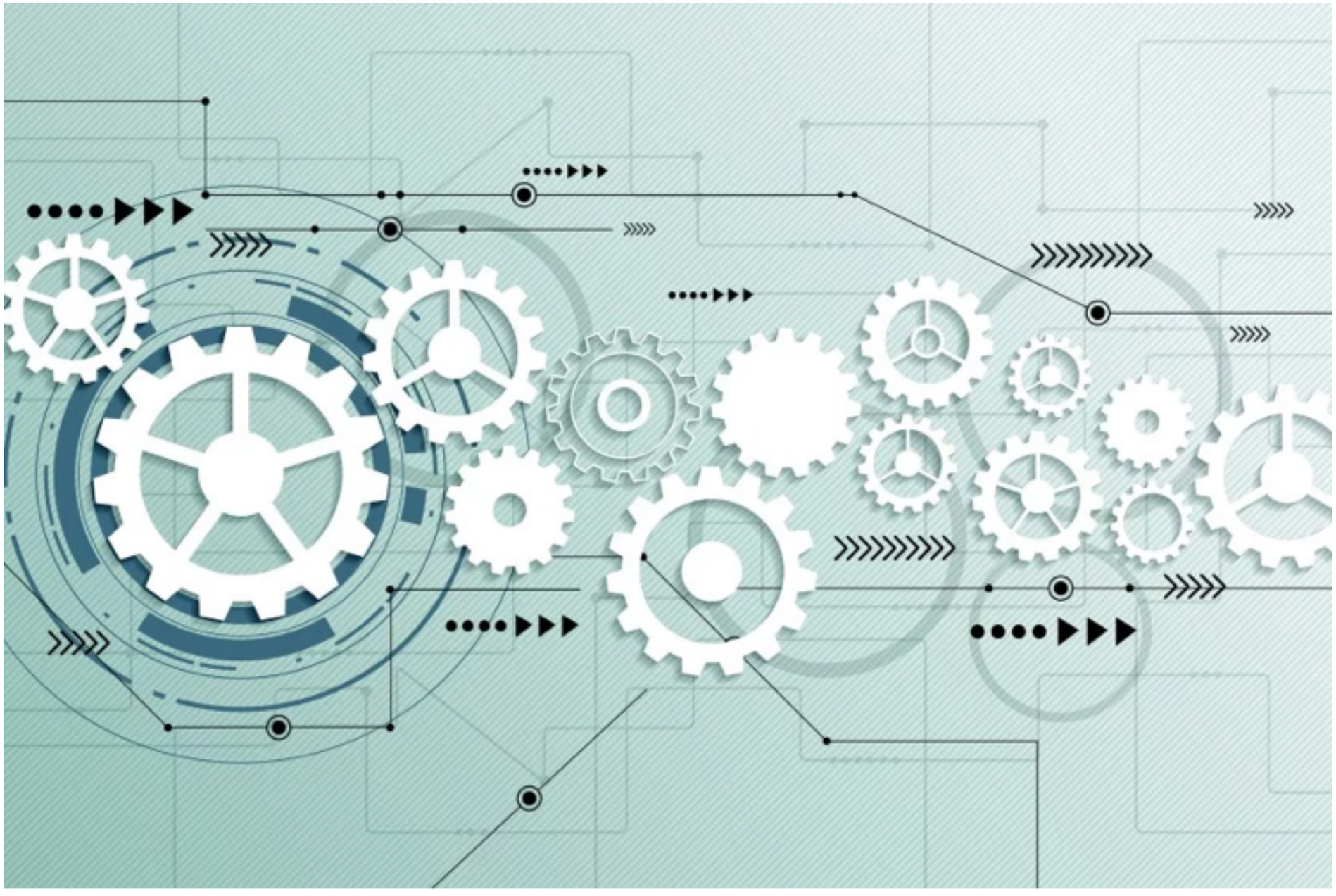
Introduction à l'assembleur ARM:

variables et assembleur



GIF-1001 Ordinateurs: Structure et Applications, Hiver 2017

Jean-François Lalonde

Source image

Accès mémoire avec variables

- Les instructions LDR et STR sont utilisées avec la syntaxe suivante pour accéder aux variables:

```
LDR Rd, maVariable   ;Met la valeur de la variable dans Rd
LDR Rd, =maVariable  ;Met l'adresse de la variable dans Rd
```

- L'assembleur traduit les lignes en plusieurs instructions du processeur

- Comment fait-il?

Valeur de la variable

Adresses	Code
	`SECTION INTVEC`
`0x0`	`B main`
	`; Déclarons une variable`
`0x4`	`maVariable DS32 1`
	`SECTION CODE`
	`main`
`0x80`	`LDR R0, maVariable`

Par quoi l'assembleur remplace-t-il `LDR R0, maVariable`?

Accès mémoire avec variables

Par quoi l'assembleur remplace-t-il `LDR R0, maVariable`?

- L'assembleur connaît déjà l'adresse de maVariable. Essayons d'aller lire le contenu mémoire à l'adresse #0x4:

```
        SECTION INTVEC

0x0     B main

        ; Déclarons une variable
0x4     maVariable DS32 1

        SECTION CODE

        main
0x80    LDR R0, [#0x4]
```

- 2 questions:
  - pouvons-nous effectuer cette opération en ARM?
  - combien de bits avons-nous besoin pour représenter une adresse?

2 questions

- Pouvons-nous effectuer cette opération en ARM?
  - NON! LDR demande toujours un registre de base…
  - Solution? Utiliser *PC*
- Combien de bits avons-nous besoin pour représenter une adresse?
  - 32. Ce qui est le même nombre que pour l'instruction elle-même… impossible de mettre une adresse de 32 bits dans une instruction de 32 bits!
  - Solution? Utiliser un déplacement *par rapport à PC*. Nous aurons besoin de moins de 32 bits pour encoder ce déplacement…

Accès mémoire avec variables

Par quoi l'assembleur remplace-t-il `LDR R0, maVariable`?

- Remplaçons par un déplacement *relatif à PC*. Déplacement de combien?
  - L'instruction est à l'adresse 0x80. Cela veut dire que PC contient 0x88.
  - La variable est à l'adresse 0x4.
- Nous voudrions que PC + *deplacement* = 0x4.
  - Donc, *deplacement* = 0x4 - PC = 0x4 - 0x88 = -0x84.

```
        SECTION INTVEC

0x0     B main

        ; Déclarons une variable
0x4     maVariable DS32 1

        SECTION CODE

        main
0x80    LDR R0, [PC, #-0x84]
```

Attention!
Le déplacement maximal permis est de 4096 octets (0x1000).

Démonstration

(Accès mémoire avec variable (1))

Adresse de la variable

Adresses	Code
	`SECTION INTVEC`
`0x0`	`B main`
	`; Déclarons une variable`
`0x4`	`maVariable DS32 1`
	`SECTION CODE`
	`main`
`0x80`	`LDR R0, =maVariable`

Par quoi l'assembleur remplace-t-il `LDR R0, =maVariable`?

Accès mémoire avec variables

Par quoi l'assembleur remplace-t-il `LDR R0, =maVariable`?

- L'assembleur connaît déjà l'adresse de maVariable. Essayons de remplacer maVariable par son adresse:

```
                SECTION INTVEC
0x0             B main

                ; Déclarons une variable
0x4             maVariable DS32 1

                SECTION CODE

                main
0x80            MOV R0, #0x4
```

- Question:
  - Combien de bits avons-nous besoin pour représenter une adresse?

2 questions

- Combien de bits avons-nous besoin pour représenter une adresse?
  - 32. Ce qui est le même nombre que pour l'instruction elle-même… impossible de mettre une adresse de 32 bits dans une instruction de 32 bits!
    - Solution? Utiliser un déplacement par rapport à PC. Nous aurons besoin de moins de 32 bits pour encoder ce déplacement…
- Cependant… où se trouve l'adresse de maVariable?
  - Nulle part! Il faut que l'assembleur l'écrive à qq part en mémoire
- Où peut-il l'écrire?

Où écrire l’adresse…

Où l’assembleur peut-il écrire l’adresse de maVariable?

```
        SECTION INTVEC

0x0     B main

        ; Déclarons une variable
0x4     maVariable DS32 1
0x8     ; ici?

        SECTION CODE

        main
        ; ou encore ici?
0x80    MOV R0, #0x4
0x84    ; ici?
```

- Typiquement, l’assembleur rajoute l’adresse de la variable *après* le code.

Accès mémoire avec variables

Par quoi l'assembleur remplace-t-il `LDR R0, maVariable`?

- Plaçons l'adresse de la variable *après* le code, soit à l'adresse 0x84.
- Remplaçons par un déplacement *relatif à PC*. Déplacement de combien?
  - L'instruction est à l'adresse 0x80. Cela veut dire que PC contient 0x88.
  - L'adresse de variable est à l'adresse 0x84.

- Nous voudrions que PC + *deplacement* = 0x84.
  - Donc, *deplacement* = 0x84 - PC = 0x84 - 0x88 = -0x4

```
                SECTION INTVEC

0x0             B main

                ; Déclarons une variable
0x4             maVariable DS32 1

                SECTION CODE

                main
0x80            LDR R0, [PC, #-0x4]
0x84            0x00000004
```

Démonstration

(Accès mémoire avec variable (2))