IUT Béziers
Département Réseaux et Télécoms
B.U.T. R&T

2021/2022
R106
S.D.

R106 – Architecture des ordinateurs

Premières notions d'assembleur

TD 3

1 Quelques informations préliminaires ...

Nous allons aujourd'hui nous initier à l'assembleur sur un processeur d'architecture ARM. Les ARM sont des processeurs simples, qui utilisent une représentation **petit-boutiste**, et possèdent un jeu d'instruction réduit. Il s'agit ici d'un processeur **32 bits**

Les ARM possèdent 16 registres, numérotés de R0 à R15. Parmi ces 16 registres,

- 13 sont des registres généraux (R0 à R12)
- 3 sont des registres spéciaux :
  - R13 (aussi nommé SP) - Stack Pointer
  - R14 (aussi nommé LR) - Link Register
  - R15 (aussi nommé PC) - Program Counter

Nous verrons plus tard le rôle de ces trois registres, mais nous pouvons d'ores et déjà noter que PC contient en général l'adresse de l'instruction située deux lignes plus loin. Ceci provient du fait que sur les anciens ARM, l'UC du processeur préchargeait deux instructions d'avance.

Comme vu en cours, les processeurs ont en général un registre de status, indiquant son état. Celui-ci ne fait pas exception et possède un registre CPSR (Current Program Status Register) de 32 bits.

En voici le détail :

```
Bits      Name    Function
          [31]    N   Negative condition code flag
          [30]    Z   Zero condition code flag
          [29]    C   Carry condition code flag
          [28]    V   Overflow condition code flag
          [27]    Q   Cumulative saturation bit
          [26:25]     IT[1:0] If-Then execution state bits for the Thumb IT  instruction
          [24]    J   Jazelle bit
          [19:16]     GE  Greater than or Equal flags
          [15:10]     IT[7:2] If-Then execution state bits for the Thumb IT instruction
          [9]     E   Endianness execution state bit: 0 - Little-endian, 1 - Big-endian
          [8]     A   Asynchronous abort mask bit
          [7]     I   IRQ mask bit
          [6]     F   FIRQ mask bit
          [5]     T   Thumb execution state bit
          [4:0]   M   Mode field
```


Version du document
4 octobre 2021

Enfin, il faut également préciser que les CPUs ARM sont dotés d'un "barrel shifter", petit circuit permettant d'effectuer un décalage sur le second argument de toute opération.

Pour terminer cette (très) rapide présentation de notre ARM, voici la liste des instructions supportées :

Instruction	Description	Instruction	Description
MOV	Move data	EOR	Bitwise XOR
MVN	Move and negate	LDR	Load
ADD	Addition	STR	Store
SUB	Subtraction	LDM	Load Multiple
MUL	Multiplication	STM	Store Multiple
LSL	Logical Shift Left	PUSH	Push on Stack
LSR	Logical Shift Right	POP	Pop off Stack
ASR	Arithmetic Shift Right	B	Branch
ROR	Rotate Right	BL	Branch with Link
CMP	Compare	BX	Branch and eXchange
AND	Bitwise AND	BLX	Branch with Link and eXchange
ORR	Bitwise OR	SWI/SVC	System Call

En plus du barrel shifter, chaque instruction peut être soumise à une condition. Voici un aperçu des différentes syntaxes que vous pourriez croiser :

```
ADD   R0, R1, R2
ADD   R0, R1, #2
MOVLE R0, #5
MOV   R0, R1, LSL #1
```

Ne vous inquietez pas, tout cela va s'éclairer petit à petit ...

2 Analyse d'un programme assembleur

Vous trouverez ci-dessous un petit programme assembleur, ainsi que l'état de la mémoire lorsque le programme est chargé et prêt à démarrer.

```
SECTION INTVEC
B main

SECTION CODE
main
MOV R0, #0x1000   ; Adresse de la première valeur
LDR R1, [R0]      ; Lecture de la première valeur dans R1
ADD R0, R0, #4    ; Adresse de la deuxième valeur
LDR R2, [R0]      ; Lecture de la deuxième valeur dans R2
ADD R1, R1, R2    ; R1 = R1 + R2
ADD R0, R0, #4    ; Adresse du résultat
STR R1, [R0]      ; Écriture du résultat en mémoire
fin
B fin

SECTION DATA
premiereValeur ASSIGN32 0x1
deuxiemeValeur ASSIGN32 0x2
resultat ALLOC32  1
```

```
    addr     00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F

0x00000000  1E  00  00  EA  00  00  00  00  00  00  00  00  00  00  00  00
0x00000010  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000020  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000030  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000040  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000050  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000060  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000070  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000080  01  0A  A0  E3  00  10  90  E5  04  00  80  E2  00  20  90  E5
0x00000090  02  10  81  E0  04  00  80  E2  00  10  80  E5  FE  FF  FF  EA
0x000000a0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x000000b0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x000000c0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x000000d0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x000000e0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x000000f0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000100  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000110  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000120  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
[...]
0x00000f80  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000f90  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000fa0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000fb0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000fc0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000fd0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000fe0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00000ff0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0x00001000  01  00  00  00  02  00  00  00  FF  FF  FF  FF  00  00  00  00
0x00001010  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
```

Exercice 1 Rappelez ce qu'est une section ? Un segment ?
Que peut contenir une section ?
Combien voyez vous de sections ici ? Quelles sont-elles ? De quel type sont-elles ?

Exercice 2 Où est stockée la section DATA ? A quelle adresse ?
Quelle différence y a t-il entre ASSIGN32 et ALLOC32 ?

Exercice 3 Où sont stockées les deux autres sections ? Quelles sont leur tailles respectives ?

Exercice 4 Expliquez en détail le fonctionnement du programme. Qu'est-ce qu'un symbole ?

Exercice 5 A quelle adresse mémoire est située la seconde ligne du main ? Sur combien d'octets est stockée cette instruction ? Qu'indique le registre PC lorsque débute l'exécution de cette instruction ?

Exercice 6 A l'aide des documents suivants, expliquez comment est traduite l'instruction MOV

Exercice 7 Ecrire un programme qui calcule l'expression $((2 * 3) + (5 * 4)$

Exercice 8 Modifiez le programme précédent de façon à faire le moins de multiplications possible.

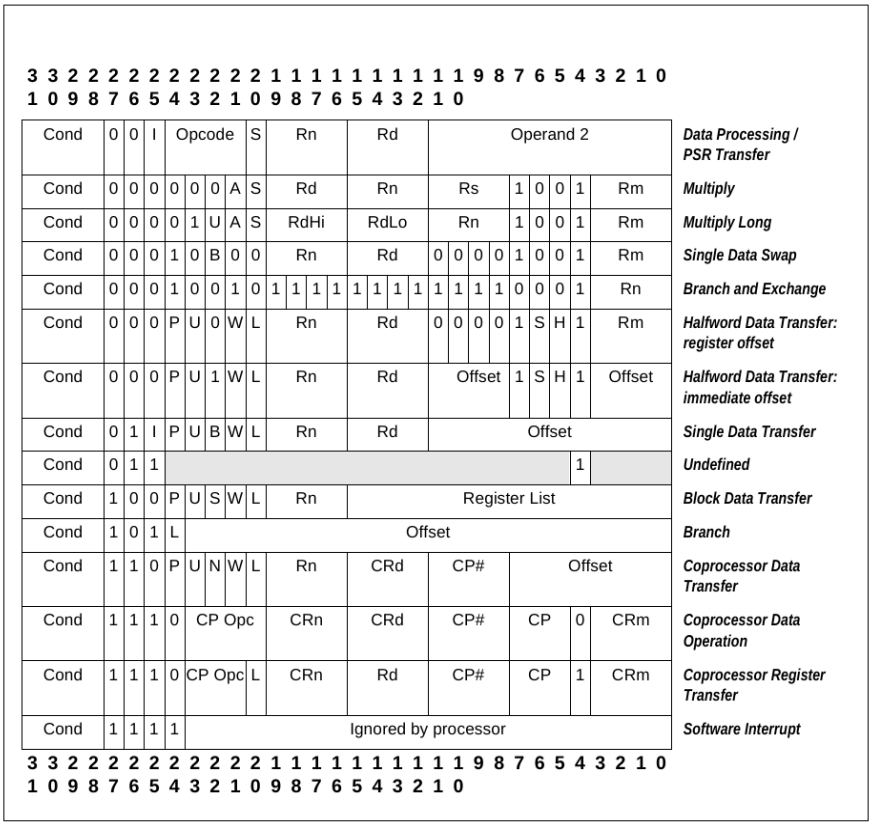

FIGURE 1 – Formats des jeux d'instructions sur ARM

Code	Suffix	Flags	Meaning
0000	EQ	Z set	equal
0001	NE	Z clear	not equal
0010	CS	C set	unsigned higher or same
0011	CC	C clear	unsigned lower
0100	MI	N set	negative
0101	PL	N clear	positive or zero
0110	VS	V set	overflow
0111	VC	V clear	no overflow
1000	HI	C set and Z clear	unsigned higher
1001	LS	C clear or Z set	unsigned lower or same
1010	GE	N equals V	greater or equal
1011	LT	N not equal to V	less than
1100	GT	Z clear AND (N equals V)	greater than
1101	LE	Z set OR (N not equal to V)	less than or equal
1110	AL	(ignored)	always

FIGURE 2 – Codes de conditions

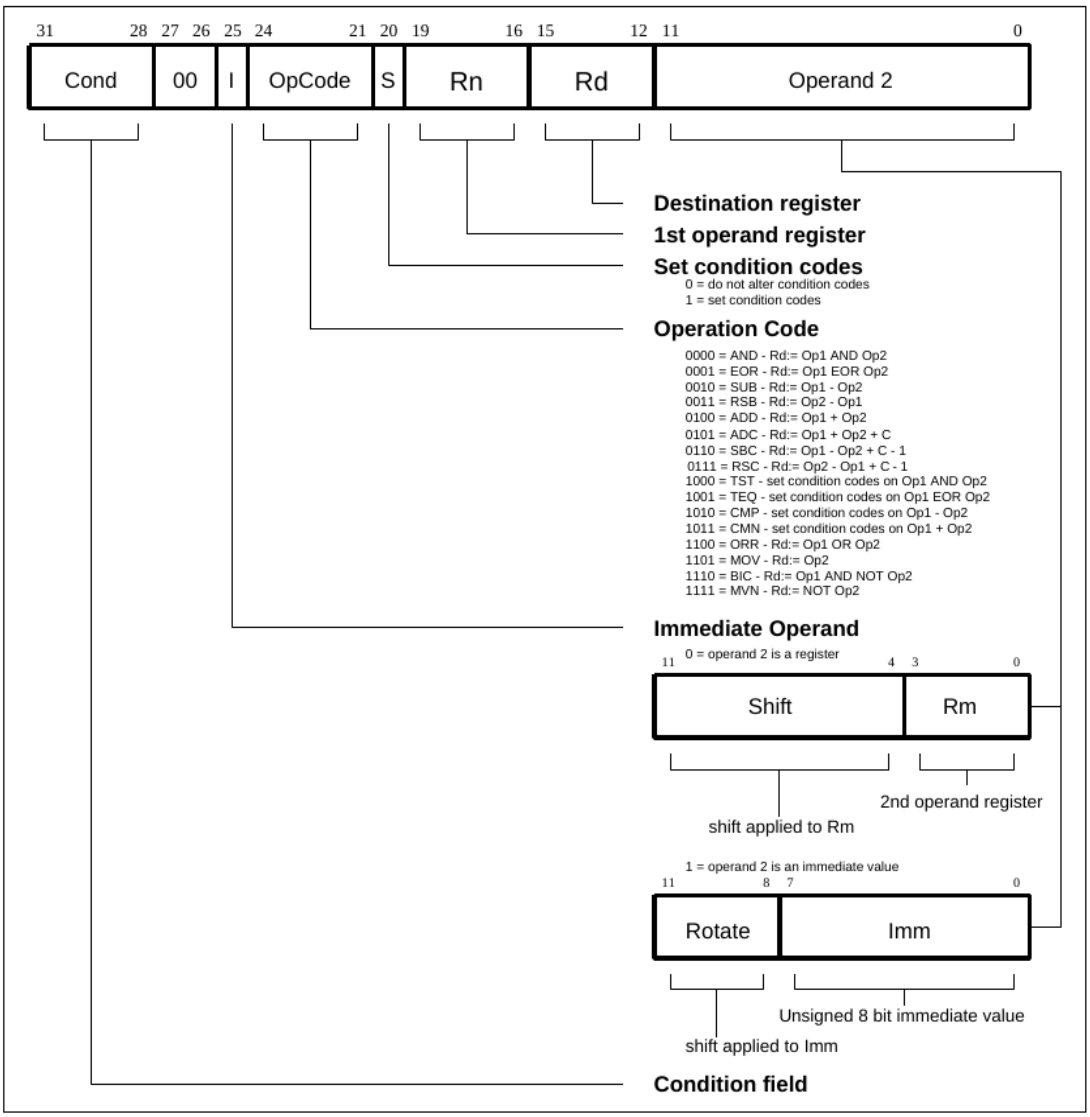

FIGURE 3 – Format de l'instruction MOV