Corrigé – exercices TD2

1. architecture – ordinateur

1.1 La taille en octets de la mémoire adressable par ce microprocesseur : 2^{32} octets.

1.2 La mémoire est de 64 kilo mots de 32 bits soit 2^{16} x 2^{2} octets, on a donc 2^{18} adresses (une adresse/octet-mémoire). L'adresse de début c'est 0x00000000 = **adr_deb**, c'est l'adresse du mot de 32 bits contenant l'octet d'adresse adr_deb, l'octet d'adresse **adr_deb+1**, l'octet d'adresse **adr_deb+2** et l'octet d'adresse **adr_deb+3**. Les adresses sont comprises dans [0,(2^{18} -1)] = [0 à 262143] = [0 à 3FFFF]. L'adresse du dernier de 32 bits : 0x0003FFFF (dernier octet), 0x0003FFFE (dernier octet -1), 0x0003FFFD (dernier octet -2), 0x0003FFFC (dernier octet -3). L'adresse du dernier mot de 32 bits est : 0x0003FFFC.

2. Structures de données de type tableau et implantation en mémoire (principale)

2.1 0x20000000 = code ascii de b = 0x62 (voir page 21 support de la séance APE1, pour le code ASCII), l'adresse de h = 0x20000000 + 18 = 0x20000012. La chaîne de caractères est : bonjour, il est 8 heures. On prend en compte les espaces. La taille du tableau (1D) = chaîne de caractères = 24 octets (un octet par caractère : y compris les espaces)

bonjour, il est 8 heures : L'adresse fin+1 = 0x20000000 + 23 +1 = 0x20000000 + 24 = 0x20000018

2.2 Le pixel de valeur 5 est stocké à l'adresse 0x20000000 +5 et le pixel de valeur 15 à l'adresse 0x20000000 +15 = 0x2000000F, avec un rangement ligne par ligne.

0,0	0,1	**0,2**	**0,3**	0	1	2	**3** → L'élément (0,3)
1,0	1,1	1,2	1,3	4	**5**	6	7
2,0	2,1	2,2	2,3	8	9	10	**11**
3,0	3,1	3,2	3,3	12	13	14	**15**

3. Ecrire en langage d'assemblage ARM Cortex – M3 les séquences suivantes :

Séquence 1 : **mode d'adresse indirect** (voir pages 19 à 23 du support de la séance APE2), l'élément (0,3) se trouve à l'adresse Adr_mem = 0x20000003 (si les pixels sont stockés en mémoire ligne/ligne)

```
Adr_mem     EQU     0x20000003
            LDR     R1,=Adr_mem     ; on charge l'adresse dans le registre R1
            LDRB    R2,[R1]         ; on charge le pixel (0.3) dans R2
            ADD     R5, #5          ; on ajoute 5 au pixel (0,3) soit 5+3=8
```

Donc R5 = 0x00000008

Faire la même séquence en changeant l'instruction LDRB R2,[R1], par un mode d'adresse indirect avec déplacement. Par exemple : LDRB R2,[R1,#3] à condition que R1 = 0x20000000. Idem pour un mode d'adressage indirect avec index, on aura par exemple LDRB R2,[R1,R4] à condition que R4 = 0x00000003 (R4 représente le registre contenant l'index) et que R1=0x20000000.

Séquence 2 : **mode d'adressage indirect avec déplacement**, l'élément (1,2) = 0x05 se trouve à l'adresse 0x20000005 soit à 0x20000000 + 5, ainsi l'offset ou déplacement = 5.

```
Adr_deb     EQU   0x20000000
            LDR   R1,=Adr_deb
            LDRB  R2,[R1,#5]
            ADDS  R2,R2                ; R2 ← R2 + R2  ce qui revient à faire R2 ←R2*2
; on range le résultat dans l'emplacement de l'élément (0.1), son adresse est = 0x20000001
            STRB  R2,[R1,#1]
```

Faire la même séquence avec le mode d'adressage indirect, et le mode d'adressage indirect avec Index.

Séquence 3 : **mode d'adressage indirect avec index**, l'élément (2,3) = 0xB soit 11 (si on considère l'image 4x4 d'origine). Cet élément se trouve à l'adresse 0x2000000B. L'index sera = à 0x0B, on choisit de le mettre dans le registre R4

```
Adr_deb     EQU   0x20000000
index       EQU   0x0B
            LDR   R1,=Adr_deb
            MOV   R4,#index
            LDRB  R2,[R1,R4]
;  Décalage logique à droite d'une position : une division par 2
            LSR   R2,#1
; On range le résultat (R2) dans l'emplacement (1,2), soit à 0x20000000 +6
            MOV   R5,#6
            STRB  R2,[R1,R5]
```