M2P CCI : Corrigé Langage Machine, Novembre 2009

1 Déclarations et if

```
      .data
      .balign 2
s:    .short 0xFFFB                  @ short s  = -5;
c:    .byte 0x6E                     @ char c = 'n';
      .balign 2
y:    .short 3                       @ unsigned short y = 3;

      .data
      .balign 2
x:    .skip 2                        @ unsigned short x;

    .text
calcul:
        @ sauvegarde des registres volontairement omise
        @ r0, r2, r4 : temporaires adresse
        @ r1, r3, r5 : temporaires données

        @ s est stocke en memoire : *&s = *&s + 1
        ldr   r0,=s                  @ s = s+1
        ldrsh r1, [r0]               @ s de type short (signed)
        add   r1, r1, #1
        strh  r1, [r0]

        ldr   r0,=x
        ldrh  r1, [r0]               @ r1 = *&x
        ldr   r2,=y
        ldrh  r3, [r2]               @ r3 = *&y

        cmp   r1, r3                 @ if (x < y)
        bhs   sinon                  @ saut a sinon si >= (entier naturel)

alors:  add  r1, r1, r2              @ *&x = *&x + *&y
        strh r1, [r0]                @ ne pas oublier de recopier en Mem

        mov  r5, #'o'                @ *&c = 'o'   mov r5,#0x6f possible
        ldr  r4,= c
        strb r5, [r4]
        bal  finsi                   @ ne pas continuer dans le sinon

sinon   add  r3, r3, #4              @ *&y = *y + 4
```

```
        strh r3, [r2]                @ ne pas oublie de recopier en Mem

finsi:  mov pc,lr                    @ ne pas oublier le branchement retour

main:   bl calcul
        mov pc,lr
```

2 Procédures, boucle et tableau

```
        .text
@ utilisation de r4 a r7 comme temporaire(s)

@ val = *adr_max    (**&adr_max)

    ldr   r4,= adr_max      @ adresse du pointeur
    ldr   r5, [r4]          @ contenu du pointeur
    ldr   r9, [r5]          @ contenu de la variable pointee
                            @ ou ldr r9,= adr_max; ldr r9,[r9]; ldr r9,[r9]

@ pt = tableau + 1         : tableau est une constante adresse
    ldr   r4,= tableau
    add   r2, r4, #4        @ adr debut de tableau + 1 * sizeof(long)
                            @ ou ldr r2,=tableau; add r2, r2, #4

@ adr_max = pt_maxi         @ *&adr_max = pt_maxi
    ldr   r4,= adr_max      @ adresse du pointeur
    str   r3, [r4]          @ affectation du contenu de pt_maxi



@ Pseudo code C equivalent a la boucle

@        borne = tableau + NB
@        goto testw
@ corps:    /* traduction du if ici */
@           pt++
@ testw:    /* comparer pt et borne */
@        si (pt < borne) goto corps

@ Rappel : on compare les adresses d'éléments de tableau
@                     et non les contenus

@ borne dans r7
        ldr r7,= tableau
```

```
        add r7, r7, #16             @ tableau + 4*sizeof(long)

        b testw                     @ goto testw

corps:  /* placer ici le code du if */
        add r2, r2, #4              @ pt++     (1*sizeof(long))

testw:  cmp r2,r7
        blo  corps                  @ if (pt < borne) goto corps
                                    @ les adresses sont des entiers
                                    @ naturels : condition < pour naturel


@ Appel de pos_max
        ldr r0,= maximum            @ max_de_pos_max (r0) = & maximum
        bl  pos_max                 @ branchement en sauvant l'adresse de retour

@ Affectation de indice
        ldr r4,= adr_max            @ adresse du pointeur
        ldr r5, [r4]                @ contenue du pointeur (r5 = *&adr_max)
        ldr r7,= tableau            @ adresse du tableau
        sub r6,r5,r7                @ différence (des adresses d'octet)
        mov r8, r6, LSR #2          @ différence (des indices) : diff_octets/4

        .ltorg
```