

Nom :

Prénom :

Grp TP :

Écrivez très lisiblement et utilisez le brouillon avant d'écrire sur la copie.

1. Définition de données

Écrivez les directives nécessaires pour définir les données selon les commentaires présents. N'oubliez pas les labels.

```
    ; tableau T3 contient 28, 20, 56, 72, 68
```

```
    ; MSG = chaîne «Bonjour» suivie d'un zéro
```

2. Arithmétique

Traduire les commentaires en instructions. Vous pouvez faire comme vous voulez. Il n'y a aucune cohérence d'un commentaire à l'autre. Toute instruction correcte rapporte des points.

```
X1:    DB ...  
X2:    DB ...  
X3:    DB ...
```

```
    ; X2 = X3 + 11
```


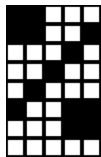
```
    ; X3 = X2 * 2
```

```
    ; X1 = X1 - X3 + X2
```

```
    ; X2 = X1 + X3 * 4
```

3. Entrées-sorties

Écrivez les instructions sous les commentaires correspondants.

<pre>V: DB ... ; écrire la valeur de la variable V sur le port d'entrées/sorties n°10 ; allumer les leds (port n°0) ainsi (rond noir=éteinte=0, blanc=allumée=1) : ; dessiner ce signe sur l'afficheur LCD (noir=1, blanc=0), port n°8 :</pre>	
	

4. Structures de contrôle

Écrivez les instructions sous les commentaires correspondants. N'oubliez pas les labels.

<pre>M: DB ... (entier non signé) W: DB ... (entier non signé) ; if (M <= W + 7) { ; M = W + 9 ; }</pre>

Écrivez les instructions sous les commentaires correspondants. N'oubliez pas les labels.
Traduire sans chercher à savoir si les algorithmes sont corrects ou non.

```
K:      DB ... (entier non signé)
        ; K = 4

        ; while (K <= 11) {

        ;     écrire la valeur de K sur le port d'entrées/sorties n°10

        ;     K = K + 3

        ; }

```

```
W:      DB ... (entier non signé)
        ; if (W > 14) {

        ;     W = W - 6

        ; } else {

        ;     W = W + 3

        ; }

```

Écrivez les instructions sous les commentaires correspondants. N'oubliez pas les labels.
Traduire sans chercher à savoir si les algorithmes sont corrects ou non.

```
C1:      DB ... (entier non signé)
C2:      DB ... (entier non signé)

; C1 = 0

; C2 = 10

; répéter {

;     if (C1 >= C2 + 1) {

;         C1 = C1 - C2

;     } else {

;         C2 = C1 + 1

;     }

; } jusqu'à (C1 > 90)
```

5. Fonctions

Traduire les commentaires en instructions. Vous pouvez faire comme vous voulez.

NB: dans ce programme, on n'utilise que les registres R0 et R1.

```
    ; R0 = 10

    ; R1 = 0

    ; while (R0 != 0) {

    ;     R1 = R1 + modifR0(R0); (voir la définition de la fonction modifR0)

    ;     R1 = R1 - 6;

    ; }

    HLT
modifR0: ; elle reçoit un paramètre dans R0, retourne son résultat dans R0 et ne doit pas modifier R1

    ; if (R0 > 26) {

    ;     R0 = R0 * 5

    ; } else {

    ;     R0 = R0 + 4

    ; }

    ; fin de la fonction modifR0
```

Définition de données

```
label:                ; définit label = adresse actuelle
                DB valeurs... ; place ces valeurs en mémoire
                RB nombre    ; réserve nombre octets non initialisés
```

Les valeurs peuvent être écrites en décimal, \$hexa, %binaire, 'texte' ou "texte".

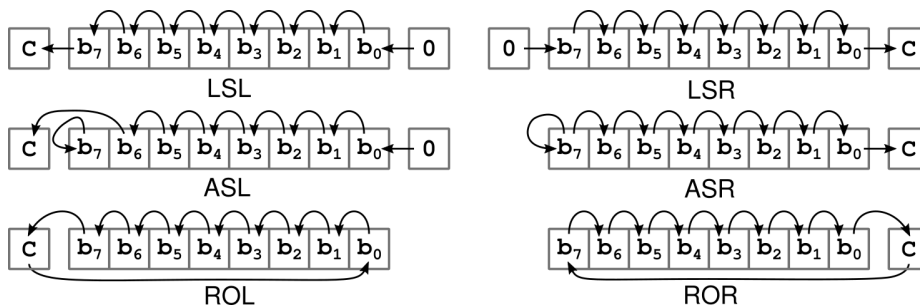
Instructions

```
LD reg, src ; affecte reg avec src | ST reg, dst ; place reg dans dst (mémoire)
```

avec

- reg (registre) : R0 ou R1
- src (source) : nombre, registre, [adresse], [registre] ou [registre+décalage]
- dst (destination) : [adresse], [registre] ou [registre+décalage]

ADD reg, src ; reg = reg + src	SUB reg, src ; reg = reg - src
ADC reg, src ; reg = reg + src + C	SBC reg, src ; reg = reg - src - C
MUL reg, src ; reg = reg × src	DIV reg, src ; reg = reg / src
CMP reg, src ; compare reg à src	MOD reg, src ; reg = reg % src
INC reg ; reg = reg + 1	DEC reg ; reg = reg - 1
NEG reg ; reg = -reg	LSL reg ; et autres, voir l'image



Sauts (si condition sur op fausse)

op	Branchement non signé	Branchement signé (convention C2 ⁸)
<	BHS (<i>Branch if Higher or Same</i>)	BGE (<i>Branch if Greater or Equal</i>)
≤	BHI (<i>Branch if HIgher</i>)	BGT (<i>Branch if Greater Than</i>)
=	BNE (<i>Branch if Not Equal</i>)	BNE (<i>Branch if Not Equal</i>)
≠	BEQ (<i>Branch if EQual</i>)	BEQ (<i>Branch if EQual</i>)
≥	BLO (<i>Branch if LOwer</i>)	BLT (<i>Branch if Less Than</i>)
>	BLS (<i>Branch if LOwer or Same</i>)	BLE (<i>Branch if Less or Equal</i>)

BRA label ; saut toujours effectué	RET ; retour d'une fonction
CALL label ; appel d'une fonction	POP reg ; dépile dans le registre
PUSH reg ; empile la valeur du registre	

Entrées/sorties

```
IN reg, port ; lit le port et affecte reg | OUT reg, port ; écrit reg sur le port
```

- port 0 : 8 leds, poids fort = led de gauche
- port 8 : afficheur LCD, 1 octet = 1 colonne à droite, poids fort = pixel du bas

- port 10 : lecture / écriture de nombres entiers