T.P. 3
Space Invaders (partie 6)

Étape 1

Réalisez le sous-programme **`IsOutOfX`** qui détermine si un bitmap sort de l'écran sur l'axe horizontal. C'est-à-dire si le bord gauche du bitmap a une abscisse négative ou si le bord droit du bitmap a une abscisse supérieure ou égale à la largeur de la fenêtre de sortie vidéo.

Entrées : **A0.L** = Adresse du bitmap.
D1.W = Abscisse en pixels du bitmap (entier signé sur 16 bits).

Sorties : **Z** renvoie *false* (0) si le bitmap ne sort pas de l'écran sur l'axe horizontal.
Z renvoie *true* (1) si le bitmap sort de l'écran sur l'axe des horizontal.

Réalisez le sous-programme **`IsOutOfY`** qui détermine si un bitmap sort de l'écran sur l'axe vertical. C'est-à-dire si le bord supérieur du bitmap a une ordonnée négative ou si le bord inférieur du bitmap a une ordonnée supérieure ou égale à la hauteur de la fenêtre de sortie vidéo.

Entrées : **A0.L** = Adresse du bitmap.
D2.W = Ordonnée en pixels du bitmap (entier signé sur 16 bits).

Sorties : **Z** renvoie *false* (0) si le bitmap ne sort pas de l'écran sur l'axe vertical.
Z renvoie *true* (1) si le bitmap sort de l'écran sur l'axe vertical.

Réalisez le sous-programme **`IsOutOfScreen`** qui détermine si un bitmap sort de l'écran. C'est-à-dire si au moins un des bords du bitmap est au-delà d'un des bords de la fenêtre de sortie vidéo.

Entrées : **A0.L** = Adresse du bitmap.
D1.W = Abscisse en pixels du bitmap (entier signé sur 16 bits).
D2.W = Ordonnée en pixels du bitmap (entier signé sur 16 bits).

Sorties : **Z** renvoie *false* (0) si le bitmap ne sort pas de l'écran.
Z renvoie *true* (1) si le bitmap sort de l'écran.

Étape 2

Réalisez le sous-programme **`MoveSprite`** qui déplace un sprite. Le déplacement se fera de façon relative. Si la nouvelle position du sprite fait sortir le sprite de l'écran, alors la position du sprite restera inchangée (la nouvelle position ne sera pas prise en compte).

Entrées : **A1.L** = Adresse du sprite.
D1.W = Mouvement relatif horizontal en pixels (entier signé sur 16 bits).
D2.W = Mouvement relatif vertical en pixels (entier signé sur 16 bits).

Sorties : **Z** renvoie *false* (0) si le sprite n'a pas été déplacé (car cela le faisait sortir de l'écran).
Z renvoie *true* (1) si le sprite a été déplacé.

Les coordonnées passées en paramètre à ce sous-programme (**D1.W** et **D2.W**) seront des coordonnées relatives : elles indiqueront le déplacement en pixels que doit effectuer le sprite. Par exemple, si les valeurs de **D1.W** et **D2.W** sont respectivement –5 et 4, alors le sprite devra se déplacer de 5 pixels vers la gauche (sa position horizontale sera décrémentée de 5) et de 4 pixels vers le bas (sa position verticale sera incrémentée de 4).

Vous testerez votre sous-programme à l'aide du programme principal suivant :

```
Main                ; Fait pointer A1.L sur un sprite.
                    lea     Invader,a1

                    ; Le déplacement sera d'un pixel vers la droite.
                    move.w  #1,d1
                    move.w  #0,d2

\loop               ; Affiche le sprite.
                    jsr     PrintSprite
                    jsr     BufferToScreen

                    ; Déplace le sprite et reboucle
                    ; jusqu'à atteindre le bord droit de l'écran.
                    jsr     MoveSprite
                    beq     \loop

                    illegal
```

Utilisez le même sprite que précédemment :

```
Invader             dc.w    SHOW                    ; Afficher le sprite
                    dc.w    0,152                   ; X = 0, Y = 152
                    dc.l    InvaderA_Bitmap         ; Bitmap à afficher
                    dc.l    0                       ; Inutilisé
```

Si tout va bien, ce programme doit afficher un envahisseur se déplaçant horizontalement de la bordure gauche de l'écran vers la bordure droite. Le programme se terminera une fois que l'envahisseur aura atteint la bordure droite.

Étape 3

L'objectif de cette étape est de déplacer un sprite à l'aide des touches du clavier. Quand la fenêtre de sortie vidéo est active, chaque touche du clavier est connectée à une case mémoire :

- Si une touche n'est pas pressée, sa case mémoire associée est mise à 00_{16} ;
- Si une touche est pressée, sa case mémoire associée est mise à FF_{16}.

La mémoire réservée pour le clavier se situe de l'adresse 400_{16} à l'adresse $4FF_{16}$.

Pour trouver l'adresse mémoire associée à une touche, vous devez :

- lancez l'émulateur ;
- affichez l'onglet mémoire ;
- sélectionnez l'adresse de départ de la vue à 400_{16} (clic droit sur la vue pour ouvrir un menu contextuel) ;
- affichez la fenêtre de sortie vidéo ;
- donnez le focus à la fenêtre de sortie vidéo ;
- appuyez sur la touche du clavier dont on souhaite connaître l'adresse mémoire associée. Le contenu de cette adresse sera alors positionné à FF_{16}.

Par exemple, l'adresse mémoire associée à la touche **[Espace]** est l'adresse 420_{16}.

Déterminez les adresses mémoire associées aux quatre touches du curseur (haut, bas, gauche, droite) et complétez la partie *Définition des constantes* de votre fichier source avec ces valeurs en respectant le modèle ci-dessous. Il sera en effet plus pratique de manipuler des étiquettes plutôt que des valeurs d'adresse.

```
                    ; Touches du clavier
                    ; ------------------------------

SPACE_KEY           equ     $420
LEFT_KEY            equ     $...
UP_KEY              equ     $...
RIGHT_KEY           equ     $...
DOWN_KEY            equ     $...
```

Réalisez le sous-programme **MoveSpriteKeyboard** qui déplace un sprite en fonction des touches du curseur. Le pas du déplacement sera d'un pixel. Votre sprite devra pouvoir se déplacer en diagonale (par exemple, si les touches **[Bas]** et **[Droite]** sont pressées simultanément).
Entrée : **A1.L** = Adresse du sprite à déplacer.

Assemblez et testez votre sous-programme à l'aide du programme principal ci-dessous. Déplacez le sprite à l'écran à l'aide des touches du curseur et vérifiez qu'ils s'arrêtent bien aux limites de l'écran (testez les quatre bords).

```
Main                ; Fait pointer A1.L sur un sprite.
                    lea     Invader,a1

\loop               ; Affiche le sprite.
                    jsr     PrintSprite
                    jsr     BufferToScreen

                    ; Déplace le sprite en fonction des touches du clavier.
                    jsr     MoveSpriteKeyboard

                    ; Reboucle.
                    bra     \loop
```