

T.P. 6 – Corrigé

Affichage Vidéo

Étape 1

```
FillScreen      ; Sauvegarde les registres dans la pile.
                 movem.l d7/a0,-(a7)

                 ; Fait pointer A0 sur la mémoire vidéo.
                 ; Cette instruction est identique à : movea.l #VIDEO_START,a0
                 lea     VIDEO_START,a0

                 ; Initialisation du compteur de boucle (D7.W).
                 ; La copie se fera sur 32 bits, c'est-à-dire sur 4 octets.
                 ; Le nombre d'itérations est donc la taille en octets divisée par 4.
                 ; Le test de sortie se fera à l'aide de DBRA,
                 ; donc D7.W doit contenir le nombre d'itérations moins 1 (cf. cours).
                 move.w  #VIDEO_SIZE/4-1,d7

\loop           ; Copie la donnée dans la mémoire vidéo
                 ; et passe à l'adresse suivante.
                 move.l  d0,(a0)+
                 dbra    d7,\loop

                 ; Restaure les registres puis sortie.
                 movem.l (a7)+,d7/a0
                 rts
```

Étape 2

```

HLines      ; Sauvegarde les registres dans la pile.
             movem.l d6/d7/a0,-(a7)

             ; Fait pointer A0 sur la mémoire vidéo.
             lea     VIDEO_START,a0

             ; D7.W = Compteur de boucle
             ;       = Nombre d'itérations - 1 (car DBRA).
             ; -----
             ; Nombre d'itérations = Nombre de rayures blanches et noires
             ; Hauteur d'un rayure blanche = 8 pixels
             ; Hauteur d'un rayure noire = 8 pixels
             ; Nombre de rayures blanches et noires = Hauteur de la fenêtre / 2x8
             move.l  #VIDEO_HEIGHT/16-1,d7

\loop        ; Dessine une rayure blanche (8 lignes blanches).
             ; -----
             ; D6.W = Compteur de boucles
             ;       = Nombre d'itérations - 1 (car DBRA)
             ; -----
             ; Nombre d'itérations = Nombre de mots longs
             ; Nombre de mots longs = Nombre d'octets / 4
             ; Nombre d'octets      = BYTE_PER_LINE x Hauteur d'une rayure blanche
             ; Hauteur d'une ligne = 8 pixels
             move.w  #BYTE_PER_LINE*8/4-1,d6
\white_loop  move.l  #$ffffffff,(a0)+
             dbra    d6,\white_loop

             ; Dessine une rayure noire (8 lignes noires).
\black_loop  move.w  #BYTE_PER_LINE*8/4-1,d6
             clr.l   (a0)+
             dbra    d6,\black_loop

             ; Reboucle tant qu'il reste des rayures
             ; blanches et noires à dessiner.
             dbra    d7,\loop

             ; Restaure les registres puis sortie.
             movem.l (a7)+,d6/d7/a0
             rts

```

Étape 3

```

WhiteSquare32    ; Sauvegarde les registres dans la pile.
                  movem.l d7/a0,-(a7)

                  ; Fait pointer A0 sur l'emplacement du carré.
                  ; -----
                  ; Centrage horizontal :
                  ; La largeur ci-dessous est mesurée en octets.
                  ; Largeur totale = Largeur de la fenêtre = BYTE_PER_LINE
                  ; Largeur du carré = 4 octets (32 pixels)
                  ; Déplacement horizontal en octets
                  ; = (Largeur totale - Largeur du carré) / 2
                  ; -----
                  ; Centrage vertical :
                  ; La hauteur ci-dessous est mesurée en pixels.
                  ; Hauteur totale = Hauteur de la fenêtre = VIDEO_HEIGHT
                  ; Hauteur du carré = 32 pixels
                  ; Déplacement vertical en pixels
                  ; = (Hauteur totale - Hauteur du carré) / 2
                  ; Déplacement vertical en octets
                  ; = Déplacement vertical en pixels x BYTE_PER_LINE
                  ; -----
                  ; Adresse du carré
                  ; = VIDEO_START + (Déplacement horizontal) + (Déplacement vertical)
                  lea    VIDEO_START+((BYTE_PER_LINE-4)/2)+(((VIDEO_HEIGHT-
                  32)/2)*BYTE_PER_LINE),a0

                  ; Initialisation du compteur de boucle (D7.W).
                  ; Nombre d'itérations = Nombre de lignes du carré (32).
                  ; D7.W = Nombre d'itération - 1 (car DBRA).
                  move.w #32-1,d7

\loop            ; Copie 32 pixels blancs dans la mémoire vidéo
                  ; et passe à l'adresse suivante.
                  move.l #$ffffffff,(a0)
                  adda.l #BYTE_PER_LINE,a0
                  dbra   d7,\loop

                  ; Restaure les registres puis sortie.
                  movem.l (a7)+,d7/a0
                  rts

```

Étape 4

```

WhiteSquare128    ; Sauvegarde les registres dans la pile.
                   movem.l d7/a0,-(a7)

                   ; Fait pointer A0 sur l'emplacement du carré.
                   ; -----
                   ; Centrage horizontal :
                   ; La largeur ci-dessous est mesurée en octets.
                   ; Largeur totale = Largeur de la fenêtre = BYTE_PER_LINE
                   ; Largeur du carré = 16 octets (128 pixels)
                   ; Déplacement horizontal en octets
                   ; = (Largeur totale - Largeur du carré) / 2
                   ; -----
                   ; Centrage vertical :
                   ; La hauteur ci-dessous est mesurée en pixels.
                   ; Hauteur totale = Hauteur de la fenêtre = VIDEO_HEIGHT
                   ; Hauteur du carré = 128 pixels
                   ; Déplacement vertical en pixels
                   ; = (Hauteur totale - Hauteur du carré) / 2
                   ; Déplacement vertical en octets
                   ; = Déplacement vertical en pixels x BYTE_PER_LINE
                   ; -----
                   ; Adresse du carré
                   ; = VIDEO_START + (Déplacement horizontal) + (Déplacement vertical)
                   lea    VIDEO_START+((BYTE_PER_LINE-16)/2)+(((VIDEO_HEIGHT-
                               128)/2)*BYTE_PER_LINE),a0

                   ; Initialisation du compteur de boucle (D7.W).
                   ; Nombre d'itérations = Nombre de lignes du carré (128).
                   ; D7.W = Nombre d'itération - 1 (car DBRA).
                   move.w #128-1,d7

\loop              ; Copie 128 pixels blancs dans la mémoire vidéo
                   ; et passe à l'adresse suivante.
                   move.l #$ffffffff,(a0)
                   move.l #$ffffffff,4(a0)
                   move.l #$ffffffff,8(a0)
                   move.l #$ffffffff,12(a0)
                   adda.l #BYTE_PER_LINE,a0
                   dbra   d7,\loop

                   ; Restaure les registres puis sortie.
                   movem.l (a7)+,d7/a0
                   rts

```

Étape 5

```

WhiteLine      ; Sauvegarde les registres dans la pile.
               movem.l d0/a0,-(a7)

               ; Nombre d'itérations = Taille de la ligne en octets
               ; D0.W = Nombre d'itérations - 1 (car DBRA)
               subq.w #1,d0

\loop          ; Copie 8 pixels blancs et passe à l'adresse suivante.
               move.b #$ff,(a0)+
               dbra   d0,\loop

               ; Restaure les registres puis sortie.
               movem.l (a7)+,d0/a0
               rts

```

```

WhiteSquare    ; Sauvegarde les registres dans la pile.
               movem.l d0-d2/a0,-(a7)

               ; D2.W = Taille en pixels du carré.
               move.w d0,d2
               lsl.w #3,d2

               ; Fait pointer A0 sur la mémoire vidéo.
               lea    VIDEO_START,a0

               ; Centre horizontalement.
               ; A0 + (Largeur totale - largeur carré) / 2
               move.w #BYTE_PER_LINE,d1
               sub.w d0,d1
               lsr.w #1,d1
               adda.w d1,a0

               ; Centre verticalement.
               ; A0 + ((Hauteur totale - Hauteur carré) / 2) * BYTE_PER_LINE
               move.w #VIDEO_HEIGHT,d1
               sub.w d2,d1
               lsr.w #1,d1
               mulu.w #BYTE_PER_LINE,d1
               adda.w d1,a0

               ; Nombre d'itérations = Taille en pixels
               ; D2.W = Nombre d'itérations - 1 (car DBRA)
               subq.w #1,d2

\loop          ; Affiche la ligne en cours et passe à la ligne suivante.
               jsr    WhiteLine
               adda.l #BYTE_PER_LINE,a0
               dbra   d2,\loop

               ; Restaure les registres puis sortie.
               movem.l (a7)+,d0-d2/a0
               rts

```