18/12/2014

Compte-rendu [MI01]

TP n°6

MOULIN MAHTIEU
LAVIOLETTE ETIENNE

1. Le cadre de pile

	Schéma de la pile (ESP)	
	31 *0*	
	… … …	
EBP+28	img_dest	*Empilement dernier paramètre*
EBP+24	img_temp2	*Empilement avant-dernier paramètre.*
EBP+20	img_temp1	*Et*
EBP+16	img_src	*Ainsi*
EBP+12	bi_height	*De*
EBP+8	bi_width	*Suite.*
EBP+4	&ret_fonction	*Adresse de retour de la fonction (suite au call)*
-> EBP=ESP	EBP	*Création cadre de pile --> Toutes les références à la pile dans le programme se feront en fonction d'EBP*
EBP-4	ebx	
EBP-8	esi	
EBP-12	edi	
EBP-16	EAX	*Sauvegarde les registres utilisés dans la fonction*
EBP-20	EBX	…
EBP-24	EDX	…
		…

MEMO : Les pop en fin de fonction suppriment respectivement EDX, EBX, EAX, edi, esi, ebx puis ebp et le return renvoie la valeur contenue dans EAX (convention C)

```
        push    ebp                     ;empilement de ebp
        mov     ebp, esp                ; ebp=esp

        push    ebx                     ; empilement de ebx
        push    esi                     ; empilement de esi
        push    edi                     ; empilement de edi

        mov     ecx, [ebp + 8]          ; ecx = bi_width
        imul    ecx, [ebp + 12]         ; ecx = bi_width * bi_height
                                        ; ecx correspond au nbre total de pixel
        mov     esi, [ebp + 16]         ; esi = img_src
        mov     edi, [ebp + 20]         ; edi = ims_temp1
```

A l'appel de la procédure, on stocke la valeur du pointeur de pile dans ebp afin d'utiliser ce registre comme référence pour lire les données stocké dans la pile, entre autres les paramètres de la fonction comme indiqué dans le cadre de pile ci-dessus. De plus, on a :

- ECX contient le nb total de pixel de l'image (height * width)
- ESI contient les pixels de l'image source
- EDI contient les pixels de l'image temporaire 1

2. Structure itératives pour parcourir l'image

2.1.Calcul des adresses source et destination

Comment faut-il initialiser EBP, ESI et EDI pour tenir compte de ces hypothèses (n'oubliez pas que chaque pixel est stocké dans un mot de 32 bits soit quatre octets) ?

```
; Code pour le compteur lignes
MOV ECX, [EBP+12]               ; ECX = height
SUB ECX,2                       ; ECX = height-2
SHL ECX, 16                     ; décalage sur la gauche : bit de poid fort = nb ligne

MOV ESI, EDI                    ; ESI = adresse img_tmp1
MOV EDI, [EBP+24]               ; EDI = adresse img_tmp2
MOV EBP, [EBP+8]                ; EBP = width
MOV EAX, EBP                    ; EAX = taille d'une ligne
IMUL EAX, 4                     ; un pixel = 4 octet
ADD EDI, EAX                    ; la premiere ligne est ignorée
ADD EDI, 4                      ; la première colonne est ignorée
```

- ECX = nb de ligne intérieur (on ne compte pas les bords)
- ESI = adresse 1er pixel de img tmp1
- EDI adresse 1er pixel de img tmp2
- EBP : Nombre de pixel par ligne

2.2.Construction de la double itération

Afin d'économiser un registre, le registre ECX va être utilisé pour réaliser à la fois le compteur de lignes i et de colonnes j. On suppose que la partie haute de ECX (16 bits de poids fort) contient le compteur de lignes restant à traiter dans l'image, la partie basse (16 bits de poids faible, accessibles en utilisant le registre CX) contient le compteur de colonnes restant à traiter dans la ligne.

2.2.1. 3.3.1 Itération sur les lignes

De quelle manière décrémenter seulement les poids forts de ECX ?

ECX sert de compteur pour les lignes et les colonnes. Les colonnes sont stockées sur les poids faible, il suffit donc de faire une soustraction standard a l'aide de sub pour décrémenter. Pour les lignes c'est légèrement différents. Le nombre de ligné étant stocké sur les 16 bits de poids forts, il faut soustraire 1 à ces bits sans modifier les bits de poids faible. Pour cela, on soustrait 10000h.

```
SUB ECX, 10000h
```

En utilisant une opération logique, comment détecter la condition d'arrêt (il ne reste plus aucune ligne à traiter) ? Indication : vous pourrez utiliser l'instruction TEST, qui modifie les drapeaux sans modifier la donnée testée.

On décrémente le compteur de ligne sur les bits de poids fort de ECX comme indiqué précédemment et on arrive à la dernière ligne lorsque celui est à 0. On effectue le test suivant :

```
TEST ECX, 0FFFF0000h
```

car on fait une soustraction juste avant.

Comment modifier ESI à la fin du traitement d'une ligne pour passer à la ligne suivante ? Même question pour EDI.

Comme on ignore les pixels sur le bord de l'image, il faut sauter deux pixel à la fin d'une ligne traitée, sur l'image 1 et l'image 2. Ainsi on ajoute 8 aux regsitre ESI et EDI pour sauter un pixel (equivalent 2 * 4 bits)

```
ADD ESI, 8
ADD EDI, 8
```

2.2.2. 3.3.2 Itération sur les colonnes

A quelle valeur initialiser CX au début de chaque ligne ?

On ajoute à ECX le nombre de pixel par ligne contenu dans EBP puis on soustrait 2 car on ignore les pixels sur les bords.

```
ADD ECX, EBP                    ; nouvelle ligne
SUB ECX, 2                      ; on ignore les bords
```

Comment doivent évoluer ESI et EDI dans cette boucle ?

ESI et EDI correspondent à l'adresse des pixels pour les images de destination 1 et 2. Ainsi, au cours de chaque itération, on leur ajoute 4 ce qui correspond à l'ajout d'un pixel à chaque fois.

Quelle devrait être la structure de l'image source et de l'image de destination après cette modification ?

2.3.3.4 Calcul du gradient de chaque pixel

On stocke la valeur de Gx dans EBX, la valeur de Gy dans EDX.

En supposant que ESI contienne l'adresse de a_{11} dans l'image source, quelle est l'adresse de a_{12} ? L'adresse de a_{21} ? L'adresse de a_{31} ?

Adresse de a_{12} = [ESI+4]

Adresse de a_{21} = [ESI+EBP*4]

Adresse de a_{31} = [ESI + EBP*8]

Implémentez un programme ce calcul de Gx et de Gy. Ne faites que les calculs nécessaires, inutile de chercher à construire un programme de convolution avec des masques quelconques.

Programme de Gx

Il s'agit de calculer la convolution d masque de Sx.

On a : $Sx = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$ et la matrice associée à l'image de départ est :

$$D = \begin{pmatrix} ESI + EBP * 8 & ESI + EBP * 8 + 4 & ESI + EBP * 8 + 8 \\ ESI + EBP * 4 & ESI + EBP * 4 + 4 & ESI + EBP * 4 + 8 \\ ESI & ESI + 4 & ESI + 8 \end{pmatrix}$$

Pour calculer la multiplication D * Sx on procède comme suit :

1. On multiplie par la colonne de gauche et on stocke le résultat dans EAX
2. On inverse le signe de EAX
3. On multiplie par la colonne de droite et on stocke le résultat dans EBX
4. On ajoute EBX a EAX
5. On fait un test pour la valeur absolue

```
MOV EAX, [ESI+4]                ; multiplication par la premiere colonne
IMUL EAX, 2                     ;
ADD EAX, [ESI]                  ;
ADD EAX, [ESI+8]                ;
NEG EAX                         ;
MOV EBX, [ESI+EBP*8+4]          ; multiplication par la troisieme colonne
IMUL EBX, 2                     ;
ADD EBX, [ESI+EBP*8]            ;
ADD EBX, [ESI+EBP*8+8]          ;
ADD EAX, EBX                    ; somme des deux
CMP EAX, 0                      ; test du signe pour la valeur absolue
JG Sy
NEG EAX
```

Programme de Gy

Il s'agit de calculer la convolution d masque de Sx.

On a : $Sx = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$ et la matrice associée à l'image de départ est :

$$D = \begin{pmatrix} ESI + EBP * 8 & ESI + EBP * 8 + 4 & ESI + EBP * 8 + 8 \\ ESI + EBP * 4 & ESI + EBP * 4 + 4 & ESI + EBP * 4 + 8 \\ ESI & ESI + 4 & ESI + 8 \end{pmatrix}$$

1. Pour calculer la multiplication D * Sx on procède comme suit :
2. On multiplie par la ligne du haut et on stocke le résultat dans EAX
3. On inverse le signe de EAX
4. On multiplie par la ligne du bas et on stocke le résultat dans EBX
5. On ajoute EBX à EAX
6. On fait un test pour la valeur absolue

```
MOV [EDI], EAX                  ; sauvegarde du calcul precedent de sx
MOV EAX, [ESI+EBP*4]            ; multiplication par la premiere ligne
IMUL EAX, 2                     ;
ADD EAX, [ESI]                  ;
ADD EAX, [ESI+EBP*8]            ;
NEG EAX                         ;
MOV EBX, [ESI+EBP*4+8]          ; multiplication par la troisieme ligne
IMUL EBX, 2                     ;
ADD EBX, [ESI+8]                ;
ADD EBX, [ESI+EBP*8+8]          ;
```

```
ADD EAX, EBX                    ; somme des deux
CMP EAX, 0                      ; test du signe pour la valeur absolue
JG calculG
NEG EAX
```

Comment calculer la valeur finale de G avant de le stocker dans l'image de destination ?

G= Gx + Gy en valeur absolue. Il ne faut pas oublier de faire un test pour vérifier que le résultat n'est pas supérieur à 255. Si c'est le cas, on lui affecte 255 comme valeur.

```
; on ajoute les deux valeurs absolues
MOV EBX, [EDI]                  ; masque Sx
ADD EAX, EBX                    ; on ajoute Sy
CMP EAX, 255                    ; test de la limite 255 max
JL trmt                         ;
MOV EAX, 255
```

3. Résultat