

## Copies et mouvements de données sur chaînes et zones écran rectangulaires

Pour ce TD on souhaite écrire un certain nombre de sous-programmes assembleurs qui pourront être appelés depuis le C (par exemple depuis le *main*) en respectant les conventions des architectures ARM pour les passages de paramètres. Les explications complètes sur les sous-programmes assembleurs sont données dans le cours de cette semaine. A ce stade on se contentera de remplir un cadre standard :

```
@ Exemple de sous-programme assembleur de prototype void asmProctest(int * arg1, int arg2);
asmProctest: @ Nom du sous-programme
push {r4-r11,lr} @ Au début du sous programme on "sauve" les valeurs de certains registres

@ Développer ici le code du sous-programme ... on a le droit d'utiliser tous les registres de r0 à r12
@ L'argument 1 est dans r0, l'argument 2 est dans r1 ... (ça marche jusqu'à 4 paramètres)

pop {r4-r11,lr} @ Là on "restaure" les valeurs initiales des registres sauvés au début
bx lr @ Fin de sous-programme : on retourne à l'appelant
```

Pour intégrer correctement un sous-programme assembleur dans un projet il est aussi nécessaire de préciser des directives techniques d'assemblage qui se trouvent avant chaque sous-programme ( `.ARM .SECTION .iwrarm,"ax",%progbits ...` ) Ces aspects techniques ne seront pas détaillés, ils varient selon l'environnement de développement.

Les noms des sous-programmes assembleurs seront préfixés par **asm**, ce n'est pas une syntaxe obligatoire mais une simple convention pour nous.

### Ecrire le code assembleur de **asmMemset** : initialiser/remplir une zone mémoire

On met la valeur *c* dans *n* octets successifs de la mémoire à partir de l'adresse pointée par *s*

```
Prototype C : void asmMemset(char * s, char c, int n);
Entrées : s = adresse de départ de la zone mémoire (r0)
c = valeur de remplissage (octet) (r1)
n = nombre d'octets à remplir (r2)
```

### Ecrire le code assembleur de **asmMemcpy** : copie de zone mémoire

On copie les *n* octets qui commencent à l'adresse *src* vers l'adresse *dest*

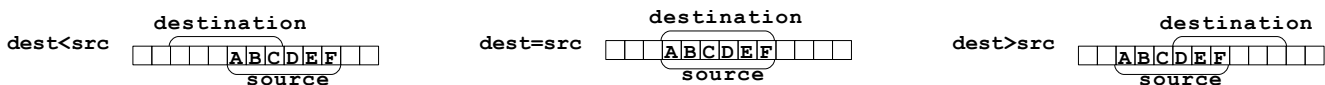
On suppose ici que les deux zones, source et destination, ne se chevauchent pas (pas d'adresses communes)

```
Prototype C : void asmMemcpy(char * dest, char * src, int n);
Entrées : dest = adresse du premier octet destination (r0)
src = adresse du premier octet source (r1)
n = nombre d'octets à copier (r2)
```

### Etudier le comportement de **asmMemcpy** en situation de chevauchement de zones

On suppose maintenant que les deux zones, source et destination, se chevauchent (adresses en commun)

Etudier, en simulant "à la main", le comportement de la fonction `asmMemcpy` précédente dans les 3 situations suivantes :



Préciser dans quel(s) cas le comportement est satisfaisant et dans quel(s) cas le comportement n'est pas satisfaisant. Par satisfaisant nous entendons : on retrouve bien dans la zone *dest*. une copie fidèle de la zone *src*. initiale.

### Ecrire le code assembleur de **asmRectcpy** : copie de zone écran rectangulaire

On copie, en VRAM, les pixels (demi-mots) d'un rectangle source vers un rectangle destination. L'adresse *src* indique l'adresse en VRAM du pixel supérieur gauche du rectangle de départ, l'adresse *dest* correspond au pixel supérieur gauche du rectangle d'arrivée. *w* et *h* sont la largeur et la hauteur de la zone rectangulaire (en nombre de pixels).

On suppose pour l'instant que les deux zones rectangulaires, source et destination, ne se chevauchent pas (pas de pixels communs)

```
Prototype C : void asmRectcpy(u16 * dest, u16 * src, int w, int h);
Entrées : dest = adresse du premier octet destination (r0)
src = adresse du premier octet source (r1)
w = largeur zone rectangulaire (r2)
h = hauteur zone rectangulaire (r3)
```

### Comportement de **asmRectcpy** en situation de chevauchement de zones : obtention d'un *scrolling*

On suppose maintenant que les deux zones rectangulaires, source et destination, vont se chevaucher (pixels en commun)

En particulier on souhaite copier le rectangle source de coordonnées (2,120) vers le rectangle destination de coordonnées (0,120) avec *w*=238 et *h*=40

On suppose toujours que les coordonnées se réfèrent au coin supérieur gauche des rectangles.

- Quelles devront être les valeurs des paramètres fournis lors de l'appel au sous-programme `asmRectcpy` depuis le C ?
- Quel sera l'effet de cette copie sur le contenu graphique (dessins...) situé sur l'écran dans les 40 lignes en bas de l'écran de la GBA ?