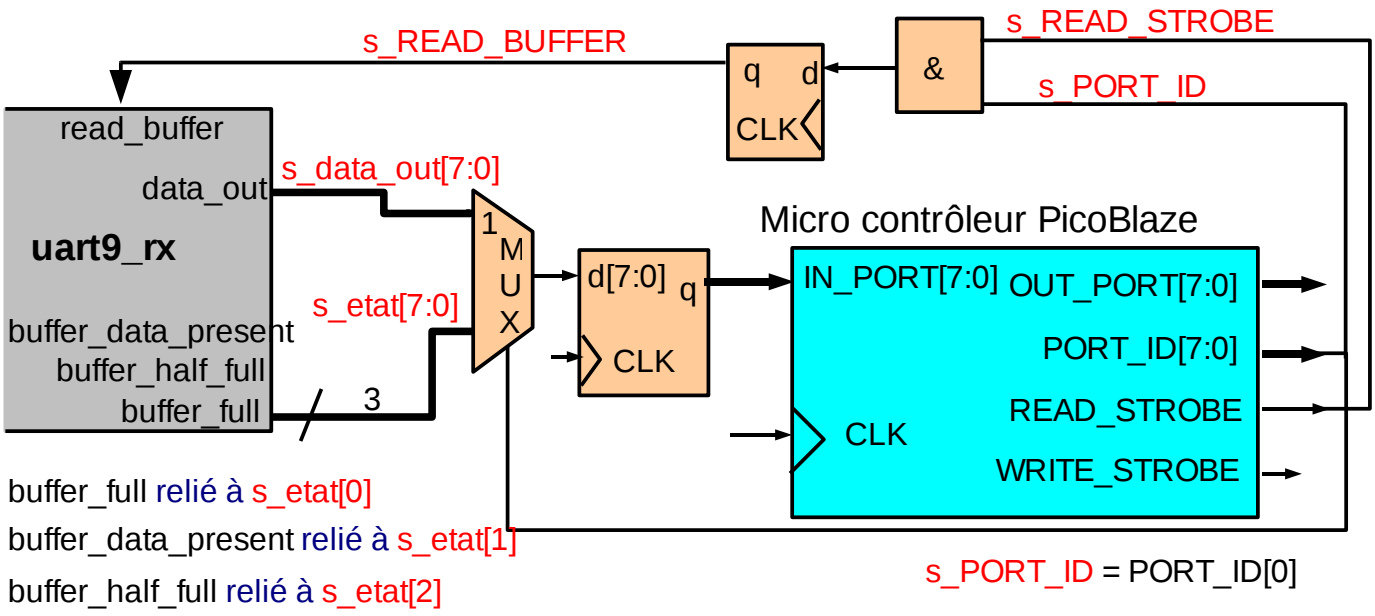
I) Introduction

Nous allons partir d'une version corrigée de l'exercice 3 du TP10 et en faire un certain nombre de modifications. Dans cette épreuve , on ne vous demande pas de vous poser des questions sur l'utilité ou non des modifications mais seulement de les exécuter et de montrer que le travail résultant fonctionne toujours. Le but de l'épreuve est ainsi d'évaluer comment vous appréhendez des modifications à partir d'un schéma complexe décrit en VHDL.
Vous disposez d'une feuille réponse sur laquelle vous répondez et sur laquelle l'enseignant notera ce qu'il a vu fonctionner et sinon, quelques indications sur l'état d'avancement de votre travail.
Une ressource "final2012.zip" vous est fournie.
Le barème n'est donné qu'à titre indicatif.

II) PicoBlaze, compteur BCD et RS232

1°) La partie matérielle vous est complètement donnée dans le répertoire exo1. Elle correspond à la correction de l'exercice 3 du TP10 (figure ci-dessous) où l'on a omis la sortie dans un registre (le seul PORT) complètement à droite.



Travail à réaliser (2 pts exo1)

2-1) Réaliser fichier ucf de l'ensemble pour que les sorties se fassent sur les afficheurs sept segments (si l'on prend le segment g en poids faible du PORT de sortie et naturellement la sélection en poids fort.

Vous testerez l'ensemble avec le programme de l'exo2 qui est donné ici tout compilé pour ce test (voir le fichier mpu_rom.vhd du répertoire exo1).

En déduire le code ASCII des caractères 'd', 'u','g' et 's' (minuscule/majuscule) qui seront utilisés par la suite.

Ressource dans **/exo1**	Tous les fichiers VHDL nécessaires
	Fichier mpu_rom.vhd contenant le programme compilé

Travail à réaliser (4 pts exo2)

2-2) Réaliser un programme qui lit les caractères qui arrivent de la RS232 et les envoie sur les afficheurs sept segments en format hexadécimal (ce que fait l'exo1). On vous donne le programme principal :

```
1               ;;; Sortie sur deux digits
2               constant MAX, A0
```

```
3               namereg s0,i
4               namereg s1,j
5               namereg s2,k
6               NAMEREG s3, octetRS232
7               NAMEREG s5, s7seg  ; rename register s5 as "s7seg"
8               NAMEREG s6, etatRS232
9               NAMEREG s7, memo
10              debut:
11                      ; initialisation RAM : table de conversion
12                      LOAD s0,7E
13                      STORE s0,00
14                      LOAD s0,30
15                      STORE s0,01
16                      LOAD s0,6D
17                      STORE s0,02
18                      LOAD s0,79
19                      STORE s0,03
20                      LOAD s0,33
21                      STORE s0,04
22                      LOAD s0,5B
23                      STORE s0,05
24                      LOAD s0,5F
25                      STORE s0,06
26                      LOAD s0,70
27                      STORE s0,07
28                      LOAD s0,7F
29                      STORE s0,08
30                      LOAD s0,7B
31                      STORE s0,09
32                      LOAD s0,77
33                      STORE s0,0A
34                      LOAD s0,1F
35                      STORE s0,0B
36                      LOAD s0,4E
37                      STORE s0,0C
38                      LOAD s0,3D
39                      STORE s0,0D
40                      LOAD s0,4F
41                      STORE s0,0E
42                      LOAD s0,47
43                      STORE s0,0F
44              boucle:
45                      ;entree des 8 bits
46                      CALL rs232 ;met lecture dans octetRS232 et mémorise dans memo
47                      AND octetRS232,0F ;on garde poids faibles
48                      FETCH s7seg,(octetRS232) ;conversion par RAM
49                      OUTPUT s7seg,0 ; sortie 7 segs
50                      CALL wait
51                      ; maintenant 4 bits poids forts
52                      LOAD octetRS232,memo
53                      SR0 octetRS232
54                      SR0 octetRS232
55                      SR0 octetRS232
56                      SR0 octetRS232
57                      AND octetRS232,0F ;inutile en principe
58                      ;conversion par RAM
59                      FETCH s7seg,(octetRS232)
60                      OR s7seg,80 ;mise a jour de la selection
61                      OUTPUT s7seg,0 ;sortie 7 segs
62                      CALL wait
63                      JUMP boucle
64
```

Ajouter donc les deux sous-programmes manquant :

- "wait" qui attend mais pas trop : une double boucle devrait suffire (inspirez-vous de l'énoncé du TP10)
- "rs232" qui a comme objectif de lire la RS232 :
    * s'il y a un caractère on lit et on stocke le résultat dans "octetRS232" et dans "memo" (déjà déclarés)
    * s'il n'y a pas de caractère on remet "memo" dans "octetRS232"

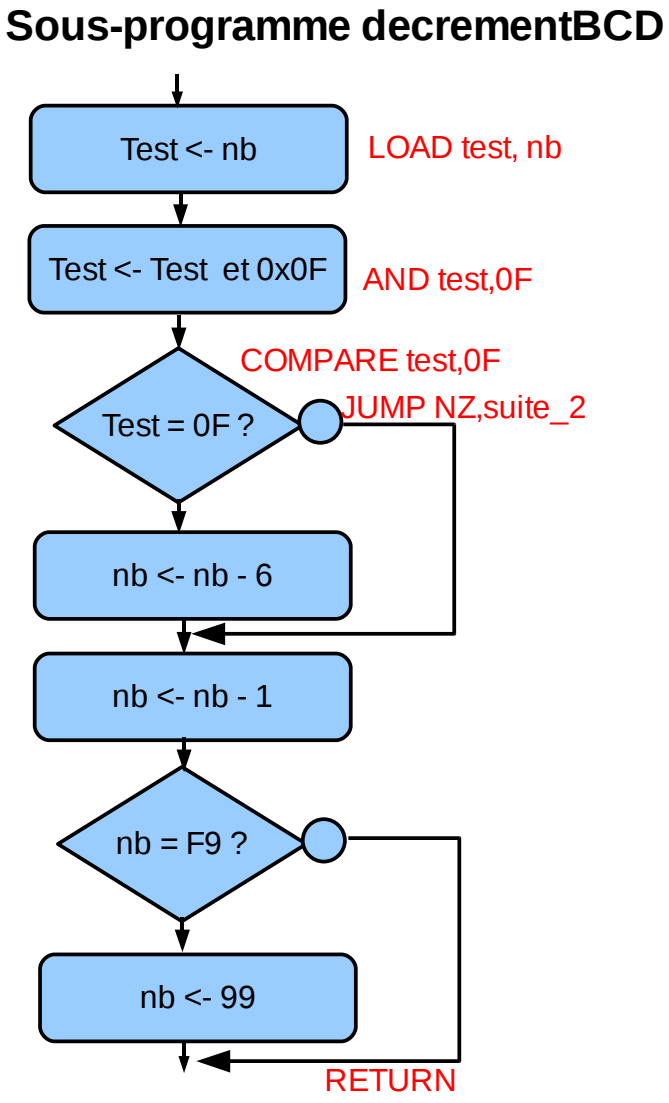
Le programme de correction de l'exercice 3 du TP10 vous est donné pour information.

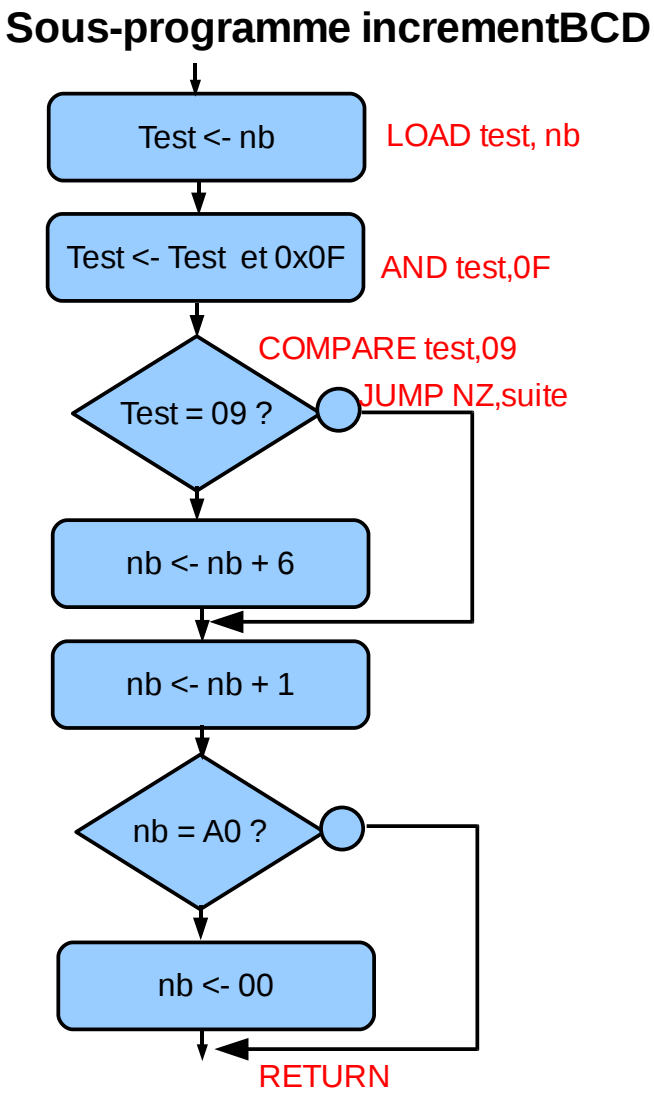
Ressource dans **/exo1**	Tous les fichiers VHDL nécessaires
Ressource dans **/exo2**	Modèle ROM_Form.vhd nécessaire pour exporter en VHDL
	TP10Exo3.psm correction de l'exercice 3 du TP 10 pour vous inspirer

Travail à réaliser (4 pts exo3)

2-3) Modifier le programme pour que l'appui sur 'd' (DOWN) décrémente un compteur et que l'appui sur 'u' (UP) l'incrémente et 's' stoppe le comptage. Tout ceci doit obligatoirement fonctionner en BCD pour une sortie correcte sur les afficheurs sept segments.

Indications : En TP11 vous aviez un algorithme capable d'incrémenter en BCD que l'on vous rappelle ci-dessous sous forme d'organigramme et on vous ajoute l'organigramme pour décrémenter.

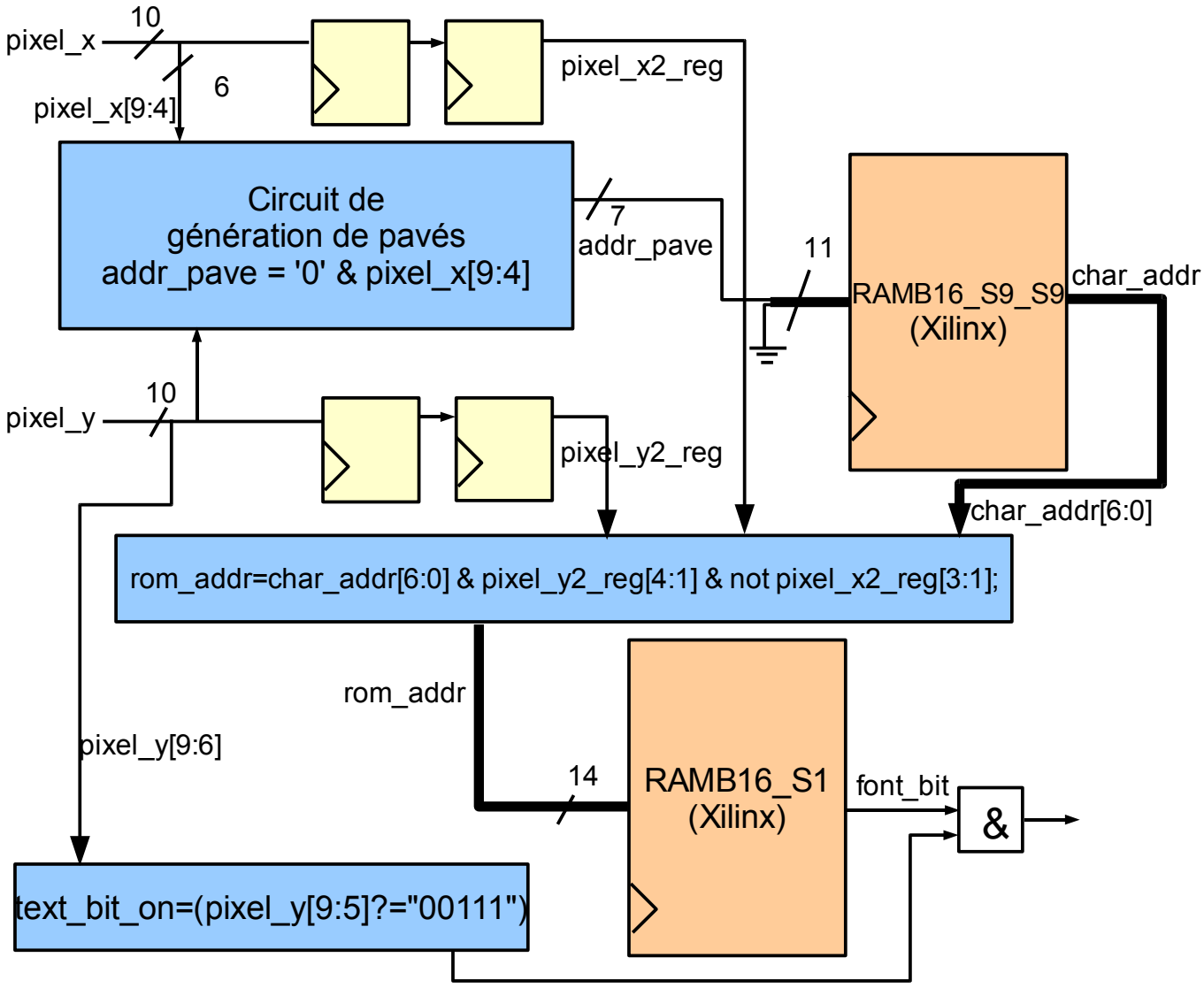




Ressource dans **/exo1**	Tous les fichiers VHDL nécessaires
Ressource dans **/exo2**	Modèle ROM_Form.vhd nécessaire pour exporter en VHDL
	TP10Exo3.psm correction de l'exercice 3 du TP 10 pour vous inspirer
Ressource dans **/exo3**	incrementBCD.psm qui est la routine qui incrémente en BCD une
	variable appelée "cmpt"

III) Préparation de l'écran VGA en mode texte

Le principe de base est réalisé par le schéma ci-dessous :



Pas de panique cet ensemble vous est donné (dans VGAStart.vhd) !
Le point essentiel est l'apparition de deux mémoires :
- une ROM de caractères (RAMB16_S1) dans laquelle on stocke la forme de ces caractères,
- une RAM vidéo (RAMB16_S9_S9) dans laquelle on stocke ce qui sera dessiné sur l'écran.

Travail à réaliser (4 pts exo4)

3-1) On vous demande de réaliser un programme top qui aura comme seul composant le programme donné et de le tester. Qu'est-ce qui est affiché sur l'écran ?

Puis remplir la RAMB16_S9_S9 pour que soit affiché :
"UTT TROYES-Final 2012-Passages : 00"
au milieu de l'écran en lieu et place de l'ancien texte. Les lettres minuscules (particulièrement le 'g') n'étant pas esthétiquement jolies vous pouvez les remplacer par des majuscules. Mais attention, pour des raisons qui vous paraitront claires plus tard, vous ne devez pas retirer ou ajouter des caractères. Ou, en tout cas, les deux zéros doivent être les 33° et 34° caractères de la ligne.
Cherchez les codes ASCII des caractères sur Internet pour ce travail. Les sorties non utilisées seront laissée open et les entrées non utilisées forcées à 0.

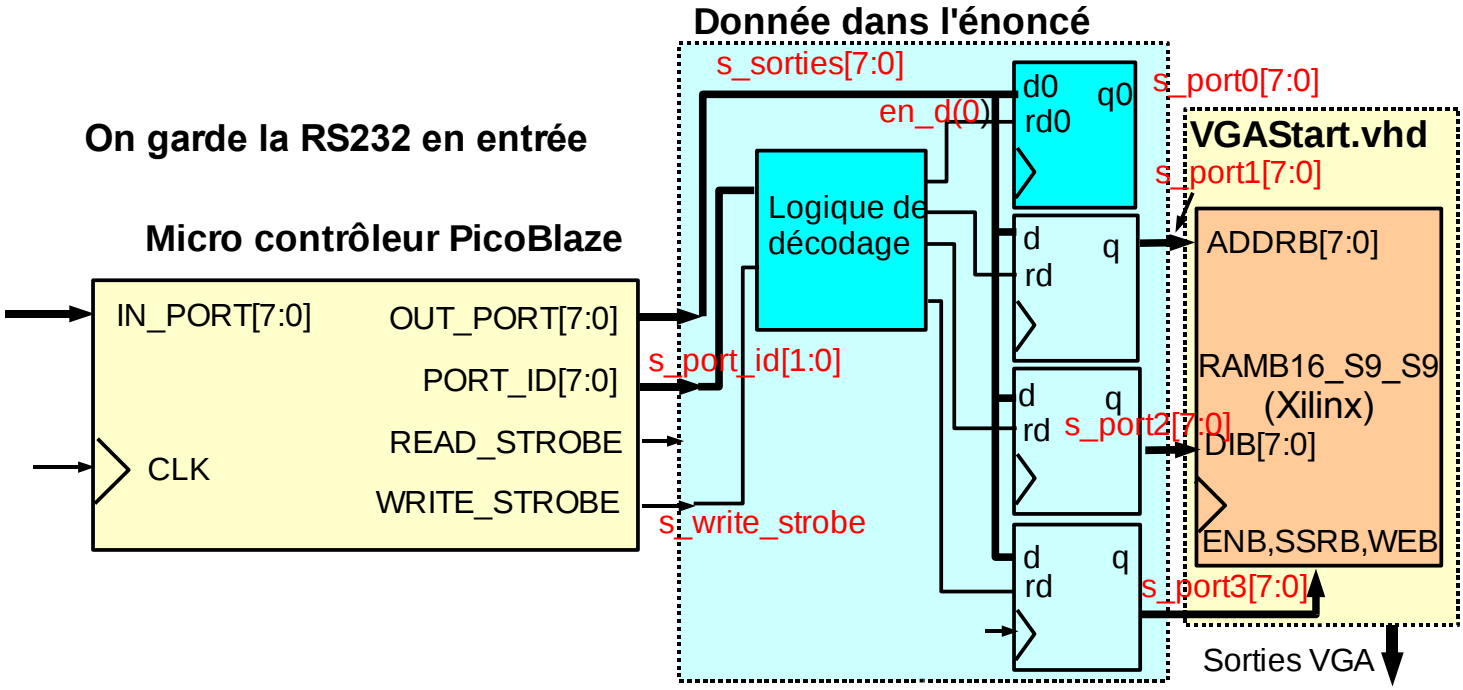
ATTENTION : la RAMB16_S9_S9 s'initialise à partir de la droite.

Ressource dans **/exo4**	Le seul fichier VHDL nécessaire à garder intact sauf initialisation de la mémoire Son nom est VGAStart.vhd

Et si l'on ajouter un picoBlaze pour achever le travail commencé.

Travail à réaliser (4 pts exo5)

3-2) On vous demande de réaliser l'interfaçage avec le picoBlaze comme indiqué ci-dessous



Les trois PORTs en blancs ajoutés sont destinées à être reliés à la mémoire RAMB16_S9_S9 comme indiqué sur la figure. Un programme tout compilé qui fonctionne comme en exo3 vous est fourni (mpu_rom.vhd).

Pour un bon fonctionnement du programme donné relier ENB au poids faible (b_0) du PORT3 et WEB au bit b_1 du PORT3.

Indications :

L'entité globale sera :

```
1             library IEEE;
2             use IEEE.STD_LOGIC_1164.ALL;
3             entity final2012_exo5 is
4               port (
5               clk,reset : in std_logic;
6               -- sorties VGA
```

```
7               hsynch,vsynch,red,green,blue : out STD_LOGIC;
8               -- RS232
9               serial_in : in std_logic
10              );
11            end final2012_exo5;
```

La logique de décodage des PORTs de sortie peut se faire avec un seul process :

```
1             -- mémorisation sorties    dans les ports
2               process(clk) begin
3               if rising_edge(clk) then
4                 if s_write_strobe = '1' then
5                   case s_port_id is
6                        when "00" => s_port0 <= s_sorties;
7                        when "01" => s_port1 <= s_sorties;
8                        when "10" => s_port2 <= s_sorties;
9                        when others => s_port3 <= s_sorties;
10                  end case;
11                end if;
12              end if;
13            end process;
```

qui nous montre que s_port_id doit être sur deux bits maintenant.

Ressource dans **/exo1**	Tous les fichiers VHDL nécessaires. Modifier final2012.vhd pour ce travail
Ressource dans **/exo4**	Le seul fichier VHDL nécessaire à garder intact sauf initialisation de la mémoire
	Son nom est VGAStart.vhd Votre fichier modifié à vous serait donc mieux
Ressource dans **/exo5**	Fichier mpu_rom.vhd contenant le programme compilé

A partir de maintenant plus aucun fichier VHDL ne sera développé : votre partie matérielle ne changera plus.

Travail à réaliser (3 pts exo6)

3-3) Modifier le sous-programme affichage du programme de l'exo 3 pour que l'ensemble fonctionne comme avec les entrées 'u', 'd' et 's' (comme dans l'exo3) mais avec la sortie sur les afficheurs de l'écran VGA. A noter que ces afficheurs ne sont plus multiplexés et que par conséquent il n'y a plus à attendre dans la routine d'affichage. Si vous voulez voir le comptage, il vous faudra par contre attendre quelque part quand même.

Ressource	La routine d'écriture en mémoire pour affichage donnée ci-dessous

```
1             ;;; examen final 2012
2             NAMEREG s5, cmpt ; compteur à afficher
3             NAMEREG s6, savecmpt ; sauvegarde du compteur pour les calculs
4             ; .............
5             ;=== routine d'affichage sur deux digits sur ecran VGA ===
6             Affichage:
7                 LOAD s0,00
8                 OUTPUT s0,3 ; bus de commande à 0
9                 LOAD savecmpt,cmpt ; sauvegarde pour la suite
10                AND cmpt,0F
11                ADD cmpt,30 ;conversion ASCII
12                LOAD octet,22 ; addresse unité
13               OUTPUT octet,1 ; adresse
14               OUTPUT cmpt,2 ; data
15               LOAD s0,03
16                OUTPUT s0,3 ; bus de commande à 3
17                LOAD s0,00
```

```
18                OUTPUT s0,3 ; bus de commande à 0
19                ; maintenant 4 bits poids forts
20                LOAD cmpt,savecmpt
21                SR0 cmpt
22                SR0 cmpt
23                SR0 cmpt
24                SR0 cmpt
25                AND cmpt,0F ;inutile en principe
26                ADD cmpt,30 ;conversion ASCII
27                LOAD octet,21 ; addresse dizaine
28                OUTPUT octet,1 ; adresse
29                OUTPUT cmpt,2 ; data
30                LOAD s0,03
31                OUTPUT s0,3 ; bus de commande à 3
32                LOAD s0,00
33                OUTPUT s0,3 ; bus de commande à 0
34                LOAD cmpt,savecmpt
35             RETURN
```

Cette routine affiche le contenu d'une variable supposée être en BCD qui s'appelle "cmpt" sans modifier sa valeur grâce à une variable auxiliaire "savecmpt" en l'écrivant à la bonne place dans la mémoire vidéo. L'écriture dans la mémoire se fait par préparation de l'adresse où l'on veut écrire (lignes 12 et 13), préparation de la donnée (en ASCII) lignes 9,10 et 11, puis enfin activation du bus de commande (lignes 7, 8, 30, 31, 32 et 33).

Cette routine ne fonctionne que si ENB est le poids faible (b_0) du PORT3 et WEB est relié au bit b_1 du PORT3.

IV) Le compteur de passages dans le picoBlaze

Il est temps de nous intéresser au but final du projet : le compteur de passage. Pour cela on garde l'architecture matérielle de la section précédente : plus de VHDL à écrire.

Travail à réaliser (4 pts exo7)

4-1) Ajouter au programme de l'exo6 un séquenceur capable d'évoluer pour faire fonctionner le compteur de passage par la liaison série avec les touches 'd' (droite) et 'g' 'gauche.

Ressource dans **/exo7**	final2011cmptPassages.psm très bien pour s'inspirer
	qui est la correction du final 2011 que l'on a fait en TP11 cette année