T.P. 2
Calculatrice (partie 1)

On désire réaliser une petite calculatrice possédant les quatre opérations de base (addition, soustraction, multiplication et division).

Exemple :
```
Veuillez saisir une expression :
50*4+2
Resultat :
202
```

La priorité des opérateurs ne sera pas prise en compte et le calcul se fera sur des nombres entiers. Chaque nombre de l'expression ne dépassera pas la valeur 32767. Le résultat affiché sera limité à 16 bits signés (une perte apparaîtra donc si le résultat théorique est supérieur).

Un message d'erreur unique ("Erreur") apparaîtra à la place du résultat si l'utilisateur saisit une expression erronée. Une expression est erronée si :

- Un des nombres est supérieur à 32767.
- L'expression contient un caractère qui n'est ni un chiffre ni un opérateur.
- Une division par zéro est apparue.
- Deux opérateurs sont côte à côte.

L'utilisateur est libre d'entrer des espaces à n'importe quel endroit de l'expression pour en améliorer la lisibilité. On suppose qu'il n'entrera jamais de zéros à gauche d'un nombre.

Ce TP s'organise en plusieurs étapes. Dans chacune d'elles, vous devrez mettre au point un sous-programme. Un fichier source unique `"TP_02.asm"` contiendra l'ensemble de tous vos sous-programmes et se divisera en quatre parties bien distinctes :

- Initialisation des vecteurs.
- Programme principal.
- Sous-programmes.
- Données.

Chaque sous-programme s'ajoutera à ce fichier source au fur et à mesure des différentes étapes. **Le programme principal devra s'adapter afin de tester le sous-programme de l'étape en cours**. À l'exception des registres utilisés pour renvoyer une valeur de sortie, aucun registre ne devra être modifié en sortie de vos sous-programmes.

– Appelez un enseignant à la fin de chaque étape pour valider votre travail –

Étape 1

Réalisez le sous-programme **`DelSpace`** qui supprime tous les espaces d'une chaîne de caractères.

Entrée : **A0.L** pointe sur le premier caractère d'une chaîne.

Exemple :											
	Avant :	`' '`	`'5'`	`' '`	`'+'`	`' '`	`' '`	`'1'`	`'2'`	`' '`	**0**
	Après :	`'5'`	`'+'`	`'1'`	`'2'`	**0**	`' '`	`'1'`	`'2'`	`' '`	`0`

Indications

- L'idéal est d'utiliser deux pointeurs sur la chaîne.
  - ➔ A0 comme pointeur source.
  - ➔ A1 comme pointeur destination.
- Il faut tester chaque caractère de la chaîne dans une boucle.
- Si le caractère est nul, sortir de la boucle.
- Si le caractère n'est pas un espace, le copier de la source vers la destination.
- Si le caractère est un espace, ne pas le copier.
- Attention à ne pas oublier le caractère nul de la chaîne destination.

Étape 2

Réalisez le sous-programme **`ErrChar`** qui détermine si une chaîne non nulle ne contient que des chiffres.

Entrée : **A0.L** pointe sur le premier caractère d'une chaîne non nulle.

Sortie : **Z** renvoie `1` si la chaîne contient au moins un caractère qui n'est pas un chiffre.
Z renvoie `0` si la chaîne ne contient que des chiffres.

Indications

- Si au moins un caractère est inférieur au caractère `'0'`, il faut renvoyer **Z** = `1`.
- Si au moins un caractère est supérieur au caractère `'9'`, il faut renvoyer **Z** = `1`.
- Pour utiliser le *flag* **Z** comme valeur de sortie, il est possible de modifier directement sa valeur en passant par le registre **CCR** (*Condition Code Register*). **CCR** est un registre 8 bits contenant les *flags* **X**, **N**, **Z**, **V** et **C** comme le montre la figure suivante :

CCR :				**X**	**N**	**Z**	**V**	**C**

Ainsi, pour positionner le *flag* **Z** à 0, sans modifier les autres *flags*, il suffit de réaliser un ET logique avec un `0` sur **Z** et un `1` sur les autres bits. Pour positionner le *flag* **Z** à `1`, on utilisera un OU logique. Ce qui donne les deux instructions suivantes :

```
andi.b #%11111011,ccr  ; Positionne le flag Z à 0
ori.b  #%00000100,ccr  ; Positionne le flag Z à 1
```

- Il est conseillé de générer deux sortie distinctes. Une que l'on pourra nommer `\true` (qui renvoie **Z** = `1`) et l'autre `\false` (qui renvoie **Z** = `0`).

Étape 3

Réalisez le sous-programme **`ErrMax`** qui détermine si le nombre que contient une chaîne est inférieur ou égal à 32767.

Entrée : **A0.L** pointe sur le premier caractère d'une chaîne **non nulle ne contenant que des chiffres**.

Sortie : **Z** renvoie `1` si le nombre que contient la chaîne est supérieur à 32767.
Z renvoie `0` si le nombre que contient la chaîne est inférieur ou égal à 32767.

Indications

- Utiliser **`StrLen`** pour faire une première comparaison sur la taille de la chaîne.
- Si la taille est inférieure à cinq caractères, le nombre est correct.
- Si la taille est supérieure à cinq caractères, le nombre est trop grand.
- Si la chaîne comporte exactement cinq caractères, alors il faut les comparer un par un.
- Pourquoi n'est-il pas possible ici d'utiliser **`Atoui`** puis de comparer la valeur numérique obtenue à 32767 ?

Étape 4

Réalisez le sous-programme **`Conv`** qui convertit une chaîne ASCII en un entier 15 bits non signés avec une gestion des erreurs.

Entrée : **A0.L** pointe sur le premier caractère d'une chaîne.

Sorties : **Z** renvoie `0` (erreur de conversion) si la chaîne ASCII :
- est nulle,
- contient au moins un caractère qui n'est pas un chiffre,
- représente un nombre supérieur à 32767.

Z renvoie `1` dans tous les autres cas (aucune erreur de conversion).
Si **Z** renvoie `0`, alors **D0.L** n'est pas modifié.
Si **Z** renvoie `1`, alors **D0.L** contient la valeur numérique du nombre.

Indications

`Conv` réalise la même opération que **`Atoui`**, mais avec une gestion des erreurs. Il faut donc d'abord valider le contenu de la chaîne, puis appeler **`Atoui`**.

Étape 5

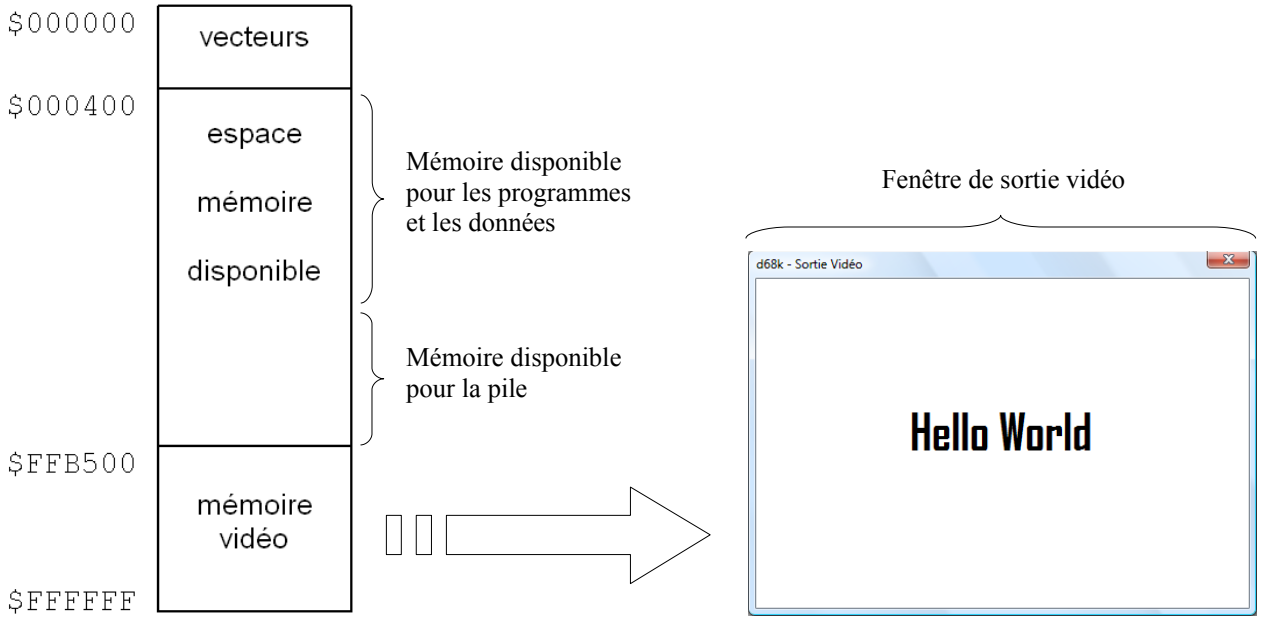
Réalisez le sous-programme **`Print`** qui affiche une chaîne de caractères dans la fenêtre de sortie vidéo du débogueur.

Entrées : **A0.L** pointe sur le premier caractère de la chaîne à afficher.
D1.B contient le numéro de colonne de la chaîne à afficher.
D2.B contient le numéro de ligne de la chaîne à afficher.

Indications

- La fenêtre de sortie vidéo du débogueur s'obtient en appuyant sur la touche **`[F4]`**.

- Le débogueur utilise une partie de la mémoire du 68000 comme mémoire vidéo. Cette zone de la mémoire est celle qui est représentée dans la fenêtre de sortie vidéo.



- Il est nécessaire de changer la position du sommet initiale de la pile. On choisira l'adresse de départ de la mémoire vidéo qui est `$FFB500`. La partie de la mémoire située juste en dessous de la mémoire vidéo sera donc réservée à la pile.

- Il est mis à votre disposition le sous-programme **`PrintChar`** qui affiche un caractère unique dans la fenêtre de sortie vidéo. Pour disposer de ce sous-programme, vous devez copier le fichier `"PrintChar.bin"` dans le même dossier que votre fichier source et inclure dans ce dernier la ligne suivante :

```
PrintChar           incbin  "PrintChar.bin"
```

`PrintChar` contient les entrées-sorties suivantes :

Entrées : **D0.B** contient le code ASCII du caractère à afficher.
D1.B contient le numéro de colonne du caractère à afficher.
D2.B contient le numéro de ligne du caractère à afficher.

Sortie : Aucune.

Il n'est pas conseillé, pour l'instant, de chercher à comprendre le fonctionnement interne du sous-programme **`PrintChar`**.

- En vous aidant de **`PrintChar`**, il suffit donc d'afficher successivement chaque caractère de la chaîne dans la fenêtre d'affichage.

- Pour cette étape, vous utiliserez le code source ci-dessous afin de tester votre sous-programme **`Print`**.

```
                    ; ==============================
                    ; Initialisation des vecteurs
                    ; ==============================
                    org     $0
vector_000          dc.l    $ffb500 ; <- Ne pas oublier l'initialisation de la pile
vector_001          dc.l    Main
vector_002_255      dcb.l   254,break_exception
break_exception     illegal

                    ; ==============================
                    ; Programme principal
                    ; ==============================
                    org     $1000

Main                lea     sTest,a0
                    move.b  #24,d1
                    move.b  #20,d2
                    jsr     Print

                    illegal

                    ; ==============================
                    ; Sous-programmes
                    ; ==============================

Print               ……      ……
                    ……      ……
                    rts

PrintChar           incbin  "PrintChar.bin"

                    ; ==============================
                    ; Données
                    ; ==============================

sTest               dc.b    "Hello World",0
```