Le Charsembleur, par -Coco-.

Ceci fonctionne comme de l'assembleur, à une chose près : chaque instruction, adresse, valeur, etc. est codée sur 1 caractère ! (C'est donc du 8 bits!)

Principe de fonctionnement :

On a 2 pointeurs : un pointeur instruction et 1 pointeur data.

Le pointeur instruction pointe sur la mémoire et permet de déterminer quelle sera la prochaine action à exécuter.

Le pointeur data pointe sur la mémoire, et il permet de la lire ou de la modifier.
L'interpréteur possède également une pile dans laquelle on peut stocker des caractères, qu'on peut dépiler par la suite.

N'oubliez pas la présence d'un flag, qui peut valoir 0 ou 1.

Pratique :

Par défaut, les deux pointeurs instruction et data pointent sur la première case du fichier, c'est à dire celle en haut à gauche.

Le fichier agit comme de la RAM. Donc si vous avez besoin de ram, prévoyez-vous un gros fichier ! (remplissez-le avec des espaces ou autres caractères par exemple)
Par exemple, si vous voulez stocker une variable à l'étiquette x, vous pouvez faire #x*, et remplacer la valeur de l'étoile par votre variable (voir le reste du manuel).

Note sur ce manuel :

f[caractère] : indique que l'instruction de base est 'f' et qu'elle prend en paramètre 'caractère', qui doit alors être directement consécutif à 'f'.
Exemple : s3; ← instruction set, vers l'étiquette 3. 3 n'est pas une instruction mains un paramètre !

Quand je parle de case suivante, il s'agit en fait de la prochaine valeur dans le programme.
De fait, s'il y a un retour à la ligne, alors la case suivante sera la première case de la ligne qui suit.
Si on se situe sur la dernière case, alors la case suivante sera la toute première case du fichier ! (celle en haut à gauche).

La syntaxe du Charsembleur :

#[caractère] : indique que le caractère qui suit immédiatement est une étiquette. Les étiquettes sont composées d'un seul caractère et permettent de donner des marqueurs dans le programme. On peut faire pointer instruction ou data vers la case qui suit directement une étiquette.

; : indique la fin de la ligne. Tout ce qui suit (ce caractère compris) ne sera même pas lu par

l'interpréteur, c'est comme si ça n'existait pas. A noter qu'il n'est absolument pas obligatoire de finir les lignes par un ;. Ce caractère sert surtout à rajouter des commentaires ! (note : doit être directement consécutif au code, pas d'espace au milieu, sinon ceux-ci seront considérés comme des instructions ! (qui n'existent pas!)

x (x étant un entier) : la valeur pure de x. x peut faire 1, 2 ou 3 caractères, et ce doit être un nombre entre 0 et 255. Par exemple, 89 est considéré comme un seul caractère dans le programme, ayant pour valeur 89 ('Y' si vous voulez tout savoir). Note pratique : 10 correspond au caractère de retour à la ligne.

**$$** : le caractère '$' (dès fois on veut l'avoir!).

Les instructions en Charsembleur (liste exhaustive) :

q : quit. Permet de quitter expressément le programme, sans aucune forme de pitié. C'est le seul moyen pour qu'un programme se quitte sans erreur.

j[etiquette] : jump. Permet de faire pointer le pointeur instruction vers la case directement après cette étiquette dans le fichier.
Exemple : jea#eq ; quittera directement sans faire l'instruction 'a'.

c[valeur] : compare la case sur laquelle pointe data avec [valeur]. Si elles sont identiques, le flag est mis à 1. Sinon, il est mis à 0.

! : non. flag := 1-flag. En gros ça inverse la valeur du flag.

k[etiquette] : saut conditionnel. Exécute un saut vers étiquette si le flag est à 1, sinon passe à l'instruction suivante. Ne remet pas le flag à 0.
Exemple global :
sxcO!kqjl; ← on va sur x. On compare x à 'O', puis on fait un non. Si le flag vaut vrai (ie x!=O) alors on saute à l'étiquette q, sinon on saute à l'étiquette l.

s[etiquette] : set. Correspond à un jump pour le pointeur data. Celui-ci pointera alors directement sur la case qui suit l'étiquette dans le fichier.

a : afficher. Permet d'afficher sur la sortie standard le caractère sur lequel pointe data.
Exemple : seaq#eA ; affiche le caractère 'A' sur la sortie standard.

d,**g**,**h**,**b** : droite, gauche, haut, bas. C'est un peu ce qui fait la particularité du charssembleur : on peut déplacer le pointeur data librement dans le fichier ! On peut donc le déplacer vers la case directement à droite, à gauche, en haut ou en bas. Là encore, c'est cyclique. Si on essaie d'accéder à la case du dessus mais que la ligne est moins longue que celle sur laquelle on était, alors data pointera vers le caractère le plus proche dans la ligne au dessus (par exemple, la dernière case si jamais on devait se retrouver à l'extérieur).
Exemple :
s3adabaq ;
#3LO ;
L ;
affichera 'L', puis 'O', puis 'L'

i : input. Écrit le caractère donnée dans l'entrée standard à l'adresse où pointe data. En gros, ça

demande un caractère et ça l'écrit dans la mémoire. (note : celui-ci utilise un getchar(). Donc vous pouvez rentrer plusieurs caractères dans l'entrée standard, il se mettront les uns à la suite des autres sur chaque i que vous trouverez. (par ex si vous avez deux i, vous pourrez rentrer 2 lettres lors du prompt, le premier i prendra la valeur de la première lettre, et le deuxième de la deuxième)) ;

l : lire. Ajoute sur la pile la valeur de la case sur laquelle pointe data.

e : écrire. Prend le dessus de la pile et l'écrit dans la case sur laquelle pointe data. Le dessus de la pile est alors détruit.

f[etiquette] : appel fonctionnel. Correspond grosso modo à un jump vers etiquette, sauf que lorsque l'instruction 'r' est exécutée, le pointeur instruction revient à la case suivant l'instruction f.
Voir l'exemple « dispchaines.prog » pour une utilisation correcte de ceci. On a maintenant possibilité de récursion !

r : retour. Instruction marquant la fin d'une fonction. Le pointeur instruction retourne alors à l'endroit où la fonction a été appelée ! (ie juste après f[c])

? : Aléatoire. La case sur laquelle pointe data prend une valeur pseudo-aléatoire !

m : mémoriser. Empile la position du pointeur data. Ceci permet de sauvegarder la position du pointeur, afin de pouvoir u retourner ensuite.

v : « va ». Depile la position de data et y retourne. Utile pour se déplacer dans de grands espaces par exemple, en utilisant de fonctions !

w[valeur] : Met la valeur [valeur] directement là où se trouve le pointeur data.
Exemple :
sxwOq#x*; ← x vaut maintenant 'O'

Arithmétique :

+ : Incrémente la valeur de la case sur laquelle pointe data. flag = 1 si on dépasse 255, 0 sinon.
- : Décrémente la valeur de la case sur laquelle pointe data. flag = 1 si on va en dessous de 0, 0 sinon.
Remarque : + et – sont circulaires : une fois dépassée la capacité on se retrouve de l'autre côté des valeurs (faire un + à 255 nous donne 0, et faire un – à 0 nous donne 255)

Allez voir la source des programmes fournis (*.prog) pour avoir des exemples plus complets !