Rapport devoir#3 IFT1227

Explications des Routines

Main : La routine principale, elle contient tous les « jal » vers les routines exigées

Saisir : Soit un utilisateur voulant entrer le texte « ah ag bt », dans la routine on initialise le registre v0 à 8 pour lire au syscall, on initialise a0 à l'Adresse en mémoire de texte (donc 268500992) et a1 contiendra 300, soit la limite de taille de l'input. Au syscall MIPS lit l'input (et s'arrêtera si on dépasse avec un 301 ième char) et nous aurons un tampon texte comme ceci :

Adresse en mémoire	Valeur ASCII du contenu
268500992	a
268500993	h
268500994	/0
268500995	a
268500996	b
268500997	/0
268500998	b
268500999	t
268501000	/0

Isalpha : La routine isalpha vérifie si un caractère donné est une lettre alphabétique (majuscule ou minuscule) selon la valeur ASCII. Elle retourne un indicateur indiquant si le caractère est valide ou non. Comme c'est une subroutine, elle est appelée non pas dans le main mais dans DecMot pour s'Assurer de « nettoyer » l'input de tout char non alphabétique. Les conditions sont simples ; les chars valides sont entre « A » et « Z », et entre « a » et « z ».

DecMots: La routine DecMot va initialiser s0 comme pointeur du début du tampon, s2 pour enregistrer les adresse des débuts de mots dans tabMots, s3 contiendra le char null et s4 sera initialisé à 0 pour compter le nombre de mots. On entre dans processTextLoop pour traiter chaque char jusqu'à ce qu'on croise le char null (la fin du string de texte). Pour chaque itération on charge le char pour l'envoyer d'abord à isalpha (vérifier si c'est un char valide) et au retour de cette subroutine si c'était un début de mot (parce qu'il était au début ou qu'il suivait un espace) on garde son adresse dans tabMot (avec le pointeur s2). On continue à traiter les chars, comme ils suivent le char du début ils font partie du même mot. Si on tombe sur un char-non alphabétique il est considéré comme une fin de mot, et on ignorera tout char non alphabétique (incluant les espaces) jusqu'à ce qu'on tombe sur un char alphabétique (pour éviter une suite de /0. Lorsqu'on croisera dans le string le char /0, c'est là qu'on arrêtera DecMots.

Afficher : La routine afficher sert à imprimer les mots du tableau tabMots et dont les chars sont dans le buffer texte. Le fait d'utiliser un adressage pour se repérer dans l'impression nous permet de réutiliser afficher même après le tri (les mêmes mots dans le même tableau, mais pas le même ordre). Avec un compteur de mots par ligne et un compteur de mot général, nous pouvons imprimer 4 mots par ligne (en s'assurant d'ajouter un espace à chaque fois qu'on croise un /0) avec le premier et savoir quand s'arrêter dans l'affichage avec le deuxième.

Trier :

Soit le texte « ah aa bt ». Basé sur notre exécution de DecMots nous aurons en mémoire ce tampon de char après traitement des char et découpage en mots

Adresse en mémoire	Valeur ASCII du contenu
268500992	a
268500993	h
268500994	/0
268500995	a
268500996	b
268500997	/0
268500998	b
268500999	t
268501000	/0

Et on aura le tableau d'index tabMots avant le tri comme ça

Adresse en mémoire	Valeur du contenu	Valeur ASCII contenue à l'adresse
268502000	268500992	a
268502004	268500993	a
268502008	268500994	b

Dans le tri initialement on a un pointeur de fin d'éléments triés pointant à l'adresse 268502012 (indiquant qu'on n'a encore rien trié, le Bubble sort étant un algorithme qui place en fin de zone de tri le maximum de chaque itération. Nous avons un pointeur pour le premier mot et pour le second (donc nous aurions un pointeur vers 269502000 et 269502004. Comme à ces adresses le contenu est une adresse aussi, nous initialisons ensuite des pointeurs (a un offset de 0, pour toujours traiter en premier le premier char) pour récupérer le contenu de ces adresse contenue dans tabMots, soit les chars. Les sub-pointeurs (utilisées dans strcmp) 268500992 et 268500993 pointent donc aux chars « a » et « a ».

Comme a et a sont identiques, nous procédons à l'incrémentation de 1 octet des sub-pointeurs (et pas les pointeurs de mots, nous travaillons avec les mêmes mots mais différents chars). On obtient

« h » et « b » On trouve qu'ils ne sont pas en bon ordre, donc nous devons les inverser dans tabMots. Comme h et b ne sont pas les chars qu'on retrouve dans tabMots (ce tableau contient les premiers chars de chaque mot, là on a traité les 2 ièmes), on récupère alors les adresses des mots avec les pointeurs de tabMots et on les échange (pas besoin de valeur temporaire dans notre code)

Ceci consiste en une itération entre 2 mots, nous traitons alors les 2 suivant en s'assurant de ne pas avoir comme 2ieme mot la valeur pointée par le pointeur de fin d'élément.

Les nouveaux mots sont donc les mots 268502004 et 268502008. Nous réinitialisons les sub-pointeurs pour obtenir les chars « g » et « b ». Ces mots doivent êtres échangés, nous récupérons donc l'adresse de tabMots (et non pas le char courant, car un même mot a plusieurs chars, mais seul le premier est utilisé pour le stockage dans tabMots). Nous continuons ainsi jusqu'à atteindre le pointeur de fin d'éléments, nous réduisons la zone de tri et continuons le tri jusqu'à ce que le pointeur de fin d 'élément triés pointe aussi au premier mot (c'est comme si on essayait de trier le premier mot, qui est lui-même déjà trié). C'est notre marqueur de fin de tri. La suite est un simple affichage