Chapitre 9

La pile

Nous allons voir une notion nouvelle permettant d'entreposer facilement les données temporaires.

9.1 Étude générale

Notion de pile.- Lorsqu'on effectue un calcul, on a souvent des résultats intermédiaires. Lorsqu'on effectue les calculs à la main, on place ces résultats intermédiaires sur un coin de la feuille. Mais comment faire lorsqu'il s'agit d'un ordinateur ?

Lorsque ces résultats intermédiaires ne sont pas en nombre très important, on peut utiliser des registres s'il y en a de libre. Mais on arrive très vite à saturation du nombre très limité de registres.

On peut alors, comme nous l'avons vu, placer ces résultats intermédiaires en mémoire vive. Reste alors à gérer la mémoire vive pour que ces résultats ne viennent pas détruire d'autres données ou le programme.

Une façon de gérer la mémoire vive consiste à réserver une certaine zone de celle-ci, appelée la **pile** (*stack* en anglais), qui sert spécifiquement à stocker les données temporaires. Son accès suit des règles spécifiques qui expliquent sa dénomination 'pile'.

Intérêt et inconvénient de la pile.- La pile sert surtout, comme nous venons de le dire, à stocker des données temporaires et, plus particulièrement, les valeurs des registres lorsqu'on est à court de registres. Une autre façon de faire serait d'augmenter le nombre de registres mais reste à savoir combien et, de toute façon, cela revient très cher. Bien entendu l'inconvénient d'utiliser une pile est qu'il s'agit de mémoire vive et donc que les temps d'accès sont importants.

Structure de la pile.- La pile est une zone connexe de la mémoire vive. Son emplacement est donc entièrement déterminé par l'adresse de début et l'adresse de fin de pile. On pourrait prévoir un index se déplaçant entre ces deux limites pour désigner l'élément qui nous intéresse.

En fait, comme son nom de pile l'indique, on accède toujours à la pile par le haut. On place un élément au sommet de la pile (on **empile**) et on récupère l'élément du sommet de la pile (on **dépile**).

L'adresse importante, à un instant donné, est donc celle du **sommet de la pile**.

Instructions d'accès à la pile.- Il y a donc deux instructions pour accéder à la pile : placer un nouvel élément (au sommet de la pile), c'est-à-dire *empiler*, et récupérer l'élément du sommet de la pile, c'est-à-dire *dépiler*.

Remarque sur la pile pleine.- Il devrait y avoir une troisième instruction pour savoir si la pile est pleine. En effet si on empile un élément alors qu'elle est pleine, on risque de détruire des données essentielles par ailleurs. Cependant cette instruction est rarement implémentée, considérant que cela a peu de chances d'arriver.

9.2 Cas du microprocesseur `i8086`

9.2.1 Mise en place

Emplacement de la pile.- Pour être sûr que le programme et la pile n'interfèrent pas, le microprocesseur `i8086` a été conçu de telle façon que le code et les données devraient être placés avec des adresses de numéro le plus bas possible dans la mémoire vive alors que la pile devrait avoir des adresses de numéro le plus haut possible. Bien entendu, l'utilisateur (ou le concepteur du système d'exploitation) peut toujours faire ce qu'il veut, mais il a intérêt à suivre cette philosophie.

Cela a pour conséquence que, pour le `i8086`, l'adresse du sommet de la pile décroit automatiquement (de deux unités) lorsqu'on ajoute un élément (de capacité un mot) à la pile et qu'elle croît (de deux unités également) lorsqu'on lui enlève un élément (de capacité un mot).

Bien entendu si la pile rencontre la portion de mémoire réservée au programme il peut se passer n'importe quoi.

Éléments de la pile.- Pour le microprocesseur `i8086`, tous les éléments de la pile sont des mots (de deux octets), ce qui explique pourquoi l'adresse du sommet de la pile croît ou décroît de deux en deux et non de un en un.

Adresse du sommet de la pile.- Cette adresse est contenue dans les **registres de pile** : le registre `SS` (pour *Stack Segment*) contient l'adresse du segment et le registre `SP` (pour *Stack Pointer*) le décalage de l'adresse du sommet de la pile.

On peut initialiser librement ces deux registres mais, dans le cas d'utilisation d'un système d'exploitation, ils sont en général initialisés par celui-ci, plus précisément par le chargeur de programme, pour une bonne gestion de la mémoire vive.

Dans tous les cas, `ss` contient lors de l'initialisation l'adresse du début de pile et `sp` la taille de la pile, ainsi `ss:sp` contient-il l'adresse de la fin de la pile.

Empilement.- Pour placer le contenu d'un registre (nécessairement de 16 bits), par exemple du registre `AX`, au sommet de la pile on utilise l'instruction :

```
push ax
```

qui place le contenu à l'adresse indiquée par `ss:sp` et décrémente `sp` de deux unités.

Dépilement.- Pour récupérer la valeur du sommet de la pile et la placer dans un registre (de 16 bits), par exemple le registre `AX`, on utilise l'instruction :

```
pop ax
```

qui a pour action de placer le contenu du sommet de pile (deux octets) dans `AX` et d'incrémenter la valeur du registre `sp` de deux unités.

Registres concernés.- Pour le `i8086` tous les registres de seize bits, sauf les registres de segment et le registre `SP`, peuvent être sauvegardés dans la pile.

Cas du registre des indicateurs.- Puisque le registre des indicateurs n'a pas de nom, il faut des instructions spéciale pour l'empiler et le dépiler. On se sert tout simplement de :

```
PUSHF
```

et de :

```
POPF
```

en se souvenant qu'indicateur se dit *flag* en anglais.

Remarque.- Si on veut conserver deux registres et les récupérer plus tard, il faut bien faire attention à les dépiler dans l'ordre inverse de leur empilement :

```
push ax
push bx
........
pop bx
pop ax
```

9.2.2 Un exemple

Le programme suivant permet d'échanger les valeurs des registres `AX` et `BX` en utilisant la pile :

```
C:>debug
-a
249C:0100 mov ax,20
249C:0103 mov bx,33
249C:0106 push ax
249C:0107 push bx
249C:0108 pop ax
249C:0109 pop bx
249C:010A int 3
249C:010B
-g
AX=0033 BX=0020 CX=0000 DX=0007 SP=FFEE BP=0000 SI=0000 DI=0000
```

```
DS=249C ES=249C SS=249C CS=249C IP=0113 NV UP EI PL NZ NA PO NC
249C:010A CC INT 3
-q
```

qui nous montre bien que les valeurs initiales de `AX` et de `BX` ont été échangées.

Exercice.- (**Initialisation du registre des indicateurs**)

Nous avons dit que nous ne pouvions pas initialiser le registre des indicateurs, contrairement aux autres registres. En fait on peut le faire, de façon détournée. Il suffit de placer la valeur que l'on veut dans un registre, d'empiler ce registre et de dépiler cette valeur dans le registre des indicateurs.