La Machine RAM

La machine RAM (*Register Addressable Memory*) est constituée de 4 éléments :

(1) Une *bande d'entrée* (E) segmentée en cases qui contiennent des entiers relatifs, accessible uniquement en *lecture* et de manière séquentielle.

(2) Une *bande de sortie* (S) segmentée en cases qui contiennent des entiers relatifs, accessible uniquement en *écriture* et de manière séquentielle.

(3) Une *mémoire* (R) indexée segmentée en *registres* R_0, R_1, ... qui contiennent des entiers relatifs. Chaque registre est accessible directement via son *adresse* spécifiée dans le *registre de sélection mémoire* (SM).

(4) Un *programme* (P), i.e. une séquence instructions codées par des couples (code opération, adresse) et indexée par le *compteur ordinal* (CO).

On charge (virtuellement) les instructions du programme dans le bloc programme et on saisit (virtuellement) les données à traiter sur la bande d'entrée.

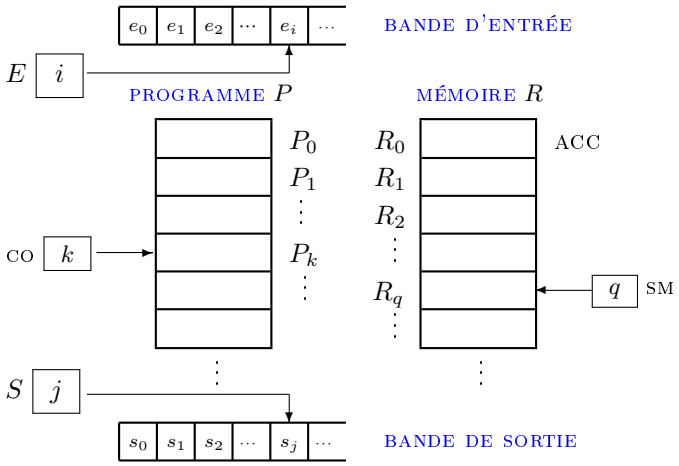

Figure 1. Description physique de la machine RAM

Ces données sont encodées sous formes d'entiers suivant un schéma d'encodage arbitraire. Les instructions $P[k]$ du programme sont décodées séquentiellement. Pour cela, le compteur ordinal contient l'adresse de la prochaine instruction à décoder, initialement 0. Après qu'une instruction a été décodée, le compteur ordinal est incrémenté pour passer à l'instruction suivante, sauf si l'instruction est une *rupture de séquence*, auquel cas son rôle est précisément de modifier la valeur du compteur ordinal. La machine s'arrête après l'instruction `STOP`.

Les données à traiter sont chargées dans la mémoire à l'aide de l'instruction de lecture `READ` qui copie le contenu de la cellule courante de la bande d'entrée dans le registre R_0 et les résultats des calculs sont renvoyés séquentiellement sur la bande de sortie à l'aide de l'instruction d'écriture `WRITE`.

Type	Instruction	Signification
Entrées/Sorties	`READ`	`ACC` $\leftarrow e_i$, $i \leftarrow i+1$
	`WRITE`	$s_j \leftarrow$ `ACC`, $j \leftarrow j+1$
Affectations	`LOAD #n`	`ACC` $\leftarrow n$
	`LOAD n`	`ACC` $\leftarrow R[n]$
	`LOAD @n`	`ACC` $\leftarrow R[R[n]]$
	`STORE n`	$R[n] \leftarrow$ `ACC`
	`STORE @n`	$R[R[n]] \leftarrow$ `ACC`
Arithmétiques	`ADD n`	`ACC` $\leftarrow$ `ACC` $+ R[n]$
	`SUB n`	`ACC` $\leftarrow$ `ACC` $- R[n]$
	`MUL n`	`ACC` $\leftarrow$ `ACC` $\times R[n]$
	`DIV n`	`ACC` $\leftarrow$ `ACC` $\div R[n]$
	`MOD n`	`ACC` $\leftarrow$ `ACC` $\%\ R[n]$
	`INC n`	$R[n] \leftarrow R[n]+1$
	`DEC n`	$R[n] \leftarrow R[n]-1$
Ruptures de séquence	`JUMP n`	`CO` $\leftarrow n$
	`JUMZ n`	`CO` $\leftarrow n$ si `ACC` $= 0$
	`JUML n`	`CO` $\leftarrow n$ si `ACC` < 0
	`JUMG n`	`CO` $\leftarrow n$ si `ACC` > 0
	`STOP`	arrêt du programme
	`NOP`	No Operation

Table 1. Instructions de la machine RAM

On dispose de l'ensemble des registres R_i de la mémoire pour y stocker des résultats (le contenu du registre R_i est noté $R[i]$). Le registre R_0 a un statut particulier, c'est l'*accumulateur* (ACC). Une opération arithmétique remplace le contenu de l'accumulateur par le résultat de l'opération entre l'accumulateur et le contenu d'un registre. La plupart des ruptures de séquence dépendent du contenu de ce registre.

À l'exception des instructions `READ`, `WRITE`, `STOP` et `NOP`, une instruction est un couple `CODOP adr` constitué d'un *code opération* et d'une *adresse*. L'adresse est soit celle d'une cellule de la mémoire, soit celle d'une cellule du programme dans le cas d'une rupture de séquence.

L'instruction `NOP` ne fait "rien". Elle peut être insérée entre des instructions du programme en prévention pour "recaler" des sauts en cas de modification du code.

Le tableau 1 regroupe les différentes instructions possibles. Les opérations arithmétiques se déclinent en adressage absolu et en adressage relatif avec `#n` et `@n` pour travailler respectivement avec la valeur n et la valeur $R[R[n]]$. Les adresses pour les ruptures de séquence peuvent également être relatives, par ex. l'instruction `JUMZ @3` fera sauter le programme à l'instruction dont le numéro est contenu dans le registre R_3.