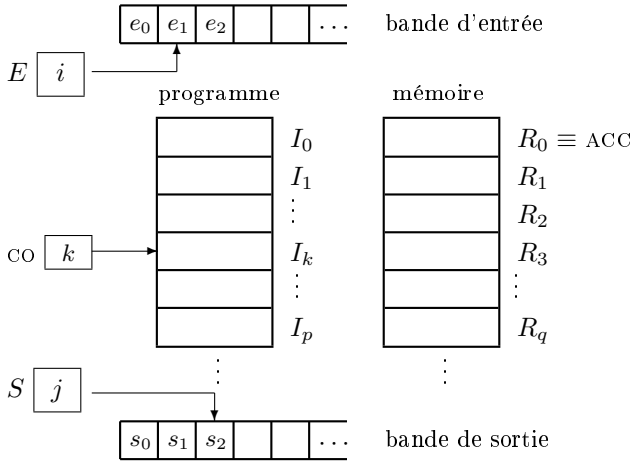
# La machine RAM

## Présentation

La machine RAM contient une suite de registres $R_0, R_1, \ldots$ constituant sa *mémoire*, d'une *bande d'entrée* indexée par un registre $E$, d'une *bande de sortie* indexée par un registre $S$ et d'un *programme* composé d'instructions $I_0,\ I_1, \ldots$ indexé par un registre CO, le *compteur ordinal*. Les registres de la mémoire et les deux bandes ne peuvent contenir que des entiers relatifs $Z$ tels que $-b \leq N < b$ où $b$ est une borne arbitrairement grande (en général une puissance de 2 par commodité). Le compteur ordinal et les registres $E$ et $S$ ne sont pas bornés.




Le cycle de fonctionnement de la machine RAM est simple : le *compteur ordinal* (CO) (initialisé à 0) contient le numéro de l'instruction à décoder. Une fois l'instruction exécutée, le compteur ordinal est incrémenté et c'est l'instruction suivante qui est exécutée et ainsi de suite sauf si l'instruction est une *rupture de séquence*. Les entrées-sorties se font respectivement sur une bande de lecture et d'écriture indexées par $\mathbf{N}$. À chaque instruction de lecture (resp. d'écriture), le registre $E$ (resp. $S$) initialement nul est incrémenté. Pour pouvoir réaliser des opérations complexes, on dispose de l'ensemble des registres $R_i$ de la mémoire pour y stocker des résultats. Le registre $R_0$ a un statut particulier, c'est l'*accumulateur*. L'exécution d'une opération arithmétique remplace le contenu de l'accumulateur par le résultat de cette opération et les instructions de rupture de séquence dépendent du contenu de ce registre.

À l'exception des instructions `READ`, `WRITE` et `STOP`, toutes les instructions sont décomposées en deux blocs, le *code opération* et l'*adresse* : `CODOP adr`. Le code indique la nature de l'opération réalisée et 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. Le tableau regroupe les différentes instructions possibles. On peut décliner les 5 opérations arithmétiques avec `#n` et `@n` pour travailler respectivement avec la valeur de $n$ et le registre $R[R[n]]$. On peut également utiliser l'adressage indirect pour l'incrément et le décrément. Pour plus de lisibilité, on note $R[n]$ le contenu du registre $R_n$. Un simulateur est accessible à l'adresse `http://zanotti.univ-tln.fr/RAM`.

| 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 |

## Algorithme de calcul de la factorielle sur la machine RAM

Ce programme de quelques lignes calcule la factorielle du nombre entier inscrit sur la première cellule de la bande d'entrée. L'hypothèse implicite sur la borne $b$ de la machine RAM utilisée est que si $n$ désigne l'entier en question, $n! < b$. Le résultat des opérations arithmétiques se faisant toujours dans l'accumulateur, il est nécessaire de conserver les résultats intermédiaires (ici les produits successifs) dans un deuxième registre $R_2$. Le registre $R_1$ contient quant à lui les valeurs successives $n$, $n-1$ etc... Il faut remarquer que l'on fait toujours le produit sans se préoccuper de la valeur du registre $R_1$ après la décrémentation. On teste alors si le produit est nul. Si c'est le cas, on affiche le contenu du registre 2, sinon on range la valeur obtenue dans le registre $R_2$.

```
ALGO FACTORIELLE
  0. READ
  1. JUMZ 8
  2. STORE 1
> 3. STORE 2
  4. DEC 1
  5. MUL 1
  6. JUMZ 11
  7. JUMP 3
> 8. LOAD #1
  9. STORE 2
 10. LOAD 2
>11. WRITE
 12. STOP
```