Définition de données

```
label:                      ; définit label = adresse actuelle
           DB valeurs...    ; place ces valeurs en mémoire
           RB nombre        ; réserve nombre octets non initialisés
```

Les valeurs peuvent être écrites en décimal, $hexa, %binaire, 'texte' ou "texte".

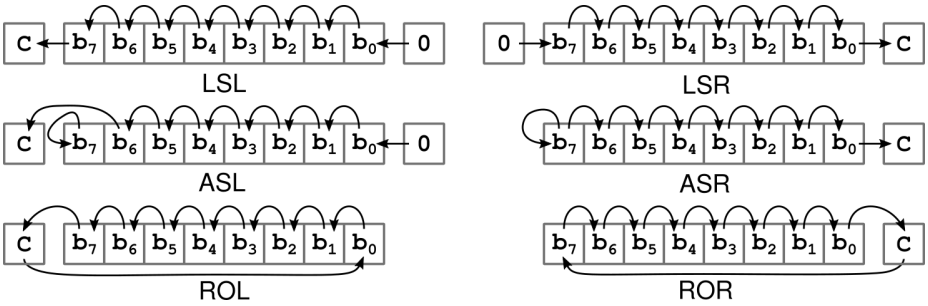
Instructions

`LD reg, src` ; affecte reg avec src	`ST reg, dst` ; place reg dans dst (mémoire)

avec

- reg (registre) : `R0` ou `R1`
- src (source) : nombre, registre, `[adresse]`, `[registre]` ou `[registre+décalage]`
- dst (destination) : `[adresse]`, `[registre]` ou `[registre+décalage]`

`ADD reg, src` ; reg = reg + src	`SUB reg, src` ; reg = reg − src
`ADC reg, src` ; reg = reg + src + C	`SBC reg, src` ; reg = reg − src − C
`MUL reg, src` ; reg = reg × src	`DIV reg, src` ; reg = reg / src
`CMP reg, src` ; compare reg à src	`MOD reg, src` ; reg = reg % src
`INC reg` ; reg = reg + 1	`DEC reg` ; reg = reg − 1
`NEG reg` ; reg = −reg	`LSL reg` ; et autres, voir l'image



Sauts (si condition sur *op* fausse)

op	Branchement non signé	Branchement signé (convention $C2^8$)
$<$	`BHS` (*Branch if Higher or Same*)	`BGE` (*Branch if Greater or Equal*)
$\leq$	`BHI` (*Branch if HIgher*)	`BGT` (*Branch if Greater Than*)
$=$	`BNE` (*Branch if Not Equal*)	`BNE` (*Branch if Not Equal*)
$\neq$	`BEQ` (*Branch if EQual*)	`BEQ` (*Branch if EQual*)
$\geq$	`BLO` (*Branch if LOwer*)	`BLT` (*Branch if Less Than*)
$>$	`BLS` (*Branch if Lower or Same*)	`BLE` (*Branch if Less or Equal*)

`BRA label` ; saut toujours effectué	
`CALL label` ; appel d'une fonction	`RET` ; retour d'une fonction
`PUSH reg` ; empile la valeur du registre	`POP reg` ; dépile dans le registre

Entrées/sorties

`IN reg, port` ; lit le port et affecte reg	`OUT reg, port` ; écrit reg sur le port

- port 0 : 8 leds, poids fort = led de gauche
- port 8 : afficheur LCD, 1 octet = 1 colonne à droite, poids fort = pixel du bas

- port 10 : lecture / écriture de nombres entiers