

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".

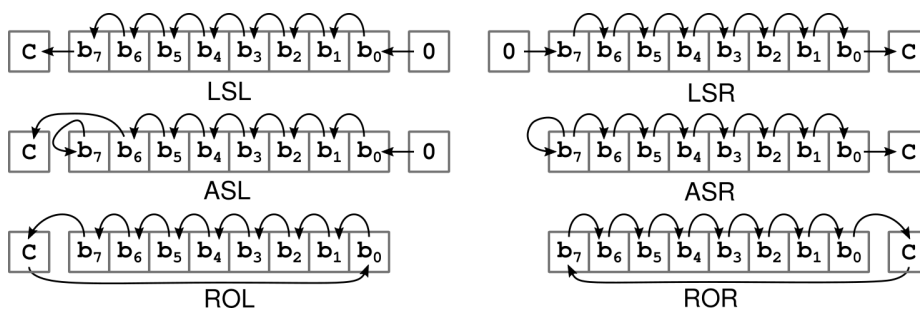
Instructions

```
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 (<i>Branch if Higher or Same</i>)	BGE (<i>Branch if Greater or Equal</i>)
≤	BHI (<i>Branch if Higher</i>)	BGT (<i>Branch if Greater Than</i>)
=	BNE (<i>Branch if Not Equal</i>)	BNE (<i>Branch if Not Equal</i>)
≠	BEQ (<i>Branch if Equal</i>)	BEQ (<i>Branch if Equal</i>)
≥	BLO (<i>Branch if Lower</i>)	BLT (<i>Branch if Less Than</i>)
>	BLS (<i>Branch if Lower or Same</i>)	BLE (<i>Branch if Less or Equal</i>)

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