Registres. [12 janvier 2023]

- Possède 4 registres d'un octet
- Registre interne: p (*registre d'état*), contient des états et codes de conditions dont *report/emprunt* (1 octet)
- Registre interne: pc (*compteur d'instruction*), contient l'adresse de la prochaine instruction (2 octets)

Nom	Utilisation principale
a	accumulateur, utilisé comme opérande et valeur de retour des opérations arithmétiques et logiques
x	utilisé comme compteur ou comme index pour l'adressage indexé
y	utilisé comme compteur ou comme index pour l'adressage indexé
s	pointeur de pile (pointe vers $0100_{16} + s$)

Valeurs immédiates.

- #: valeur numérique, sans #: adresse
- $: valeur hexadécimale
- %: valeur binaire
- Exemples:

expression	valeur
`#5`	5_{10}
`#$FF`	FF_{16}
`#%00010011`	00010011_2
`$FF`	adresse FF_{16}

Modes d'adressage.

Nom.	Syntaxe	Adresse	Exemple
absolu	`i`	i	`lda $D010`
indexé par x	`i, x`	$i + x$	`lda $D010, x`
	`etiq, x`	$etiq + x$	`lda tab, x`
indexé par y	`i, y`	$i + y$	`lda $D010, y`
	`etiq, y`	$etiq + y$	`lda tab, y`

Accès mémoire.

- Instructions, où $\text{mem}_1[a]$ dénote l'octet situé à l'adresse a de la mémoire principale:

Code d'op.	Syntaxe	Effet	Exemple
`lda`	`lda #i`	a $\leftarrow i$	`lda #42`
	`lda adr`	a $\leftarrow \text{mem}_1[adr]$	`lda var`
`ldx`	`ldx #i`	x $\leftarrow i$	`ldx #42`
	`ldx adr`	x $\leftarrow \text{mem}_1[adr]$	`ldx var`
`ldy`	`ldy #i`	y $\leftarrow i$	`ldy #42`
	`ldy adr`	y $\leftarrow \text{mem}_1[adr]$	`ldy var`
`sta`	`sta adr`	$\text{mem}_1[adr] \leftarrow$ a	`sta var`
`stx`	`stx adr`	$\text{mem}_1[adr] \leftarrow$ x	`stx var`
`sty`	`sty adr`	$\text{mem}_1[adr] \leftarrow$ y	`sty var`
`txa`	`txa`	a $\leftarrow$ x	`txa`
`tax`	`tax`	x $\leftarrow$ a	`tax`
`tya`	`tya`	a $\leftarrow$ y	`tya`
`tay`	`tay`	y $\leftarrow$ a	`tay`
`txs`	`txs`	s $\leftarrow$ x	`txs`
`tsx`	`tsx`	x $\leftarrow$ s	`tsx`
`pha`	`pha`	empile a sur la pile	`pha`
`pla`	`pla`	dépile le premier octet de la pile vers a	`pla`

Arithmétique.

Code d'op.	Syntaxe	Effet	Exemple
`adc`	`adc #i`	a $\leftarrow$ a $+ i +$ *report*	`lda #1`
	`adc adr`	a $\leftarrow$ a $+$ mem$_1[adr] +$ *report*	`adc var`
`sbc`	`sbc #i`	a $\leftarrow$ a $- i -$ *emprunt*	`sbc #1`
	`sbc adr`	a $\leftarrow$ a $-$ mem$_1[adr] -$ *emprunt*	`sbc var`
`clc`	`clc`	*report* $\leftarrow 0$ (utile avant `adc`)	`clc`
`sec`	`sec`	*emprunt* $\leftarrow 0$ (utile avant `sbc`)	`sec`
`inx`	`inx`	x $\leftarrow$ x $+ 1$	`inx`
`iny`	`iny`	y $\leftarrow$ y $+ 1$	`iny`
`inc`	`inc adr`	mem$_1[adr] \leftarrow$ mem$_1[adr] + 1$	`inc var`
`dec`	`dec adr`	mem$_1[adr] \leftarrow$ mem$_1[adr] - 1$	`dec var`

Logique.

Code d'op.	Syntaxe	Effet	Exemple
`asl`	`asl adr`	décalage logique de mem$_1[adr]$ d'un bit à gauche (directement en mémoire)	`asl var`
`lsr`	`lsr adr`	décalage logique de mem$_1[adr]$ d'un bit à droite (directement en mémoire)	`lsr var`
`and`	`and #i`	a $\leftarrow$ a $\wedge\, i$	`and #%00100011`
	`and adr`	a $\leftarrow$ a $\wedge$ mem$_1[adr]$	`and var`
`ora`	`ora #i`	a $\leftarrow$ a $\vee\, i$	`ora #%00100011`
	`ora adr`	a $\leftarrow$ a $\vee$ mem$_1[adr]$	`ora var`
`eor`	`eor #i`	a $\leftarrow$ a $\oplus\, i$	`eor #%00100011`
	`eor adr`	a $\leftarrow$ a $\oplus$ mem$_1[adr]$	`eor var`

Comparaisons et branchements.

Code d'op.	Syntaxe	Effet	Exemple
`cmp`	`cmp #i`	compare a et i	`cmp #0`
	`cmp adr`	compare a et mem$_1[adr]$	`cmp var`
`cpx`	`cpx #i`	compare x et i	`cpx #0`
	`cpx adr`	compare x et mem$_1[adr]$	`cpx var`
`cpy`	`cpy #i`	compare y et i	`cpy #0`
	`cpy adr`	compare y et mem$_1[adr]$	`cpy var`
`beq`	`beq etiq`	branche à `etiq:` si $=$	`beq boucle`
`bne`	`bne etiq`	branche à `etiq:` si $\neq$	`bne boucle`
`jmp`	`jmp etiq`	branche à `etiq:`	`jmp boucle`
`jsr`	`jsr etiq`	branche au sous-programme `etiq:` et empile l'adresse de retour	`jsr func`
`rts`	`rts`	branche à l'adresse de retour d'un sous-programme	`rts`
`rti`	`rti`	branche à l'adresse de retour d'une interruption	`rti`