

Registres.

[27 mars 2022]

- 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	<code>lda \$D010</code>
indexé par x	i, x eti _q , x	$i + x$ $eti_q + x$	<code>lda \$D010, x</code> <code>lda tab, x</code>
indexé par y	i, y eti _q , y	$i + y$ $eti_q + y$	<code>lda \$D010, y</code> <code>lda tab, y</code>

Accès mémoire.

- Instructions, où $mem_1[a]$ dénote l'octet situé à l'adresse a de la mémoire principale:

Code d'op.	Syntaxe	Effet	Exemple
<code>lda</code>	<code>lda #i</code> <code>lda adr</code>	$a \leftarrow i$ $a \leftarrow mem_1[adr]$	<code>lda #42</code> <code>lda var</code>
<code>ldx</code>	<code>ldx #i</code> <code>ldx adr</code>	$x \leftarrow i$ $x \leftarrow mem_1[adr]$	<code>ldx #42</code> <code>ldx var</code>
<code>ldy</code>	<code>ldy #i</code> <code>ldy adr</code>	$y \leftarrow i$ $y \leftarrow mem_1[adr]$	<code>ldy #42</code> <code>ldy var</code>
<code>sta</code>	<code>sta adr</code>	$mem_1[adr] \leftarrow a$	<code>sta var</code>
<code>stx</code>	<code>stx adr</code>	$mem_1[adr] \leftarrow x$	<code>stx var</code>
<code>sty</code>	<code>sty adr</code>	$mem_1[adr] \leftarrow y$	<code>sty var</code>
<code>txa</code>	<code>txa</code>	$a \leftarrow x$	<code>txa</code>
<code>tax</code>	<code>tax</code>	$x \leftarrow a$	<code>tax</code>
<code>tya</code>	<code>tya</code>	$a \leftarrow y$	<code>tya</code>
<code>tay</code>	<code>tay</code>	$y \leftarrow a$	<code>tay</code>
<code>txs</code>	<code>txs</code>	$s \leftarrow x$	<code>txs</code>
<code>tsx</code>	<code>tsx</code>	$x \leftarrow s$	<code>tsx</code>
<code>pha</code>	<code>pha</code>	empile a sur la pile	<code>pha</code>
<code>pla</code>	<code>pla</code>	dépile le premier octet de la pile vers a	<code>pla</code>

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