Ukaz LDA

***LDA*Load Accumulator**

Operacija: ACCX <-- (M)

Določen akumulator dobi vrednost iz pomnilnika

Opis:*Naloži vsebino pomnilnika v akumulator.*
Zastavice se postavijo glede na podatek.

Zastavice:

S	X	H	I	N	Z	V	C
-	-	-	-	⇕	⇕	0	-

N in Z se postavita ali ne, V je vedno 0 (ni postavjena)

N R7
Se postavi, če je MSB v rezultatu 1; sicer N=0.

Z R7' • R6' • R5' • R4' • R3' • R2' • R1' • R0'
Postavi se, če je rezultat enak $00, sicer pa ne.

V 0
Ni postavljena

Oblike: *LDAA (opr); LDAB (opr)*

Obstajata dve obliki LDA ukaza, ena za vsak akumulator

LDA

Načini naslavljanja, Strojni ukazi, Izvajanje po ciklih:

	LDAA #ii			*LDAA $dd*			*LDAA $hhll*			*LDAA (IND,X)*			*LDAA (IND,Y)*		
	Addr	*Data*	*R/W*	*Addr*	*Data*	*R/W*	*Addr*	*Data*	*R/W*	*Addr*	*Data*	*R/W*	*Addr*	*Data*	*R/W*
1	OP	86	1	OP	96	1	OP	B6	1	OP	A6	1	OP	18	1
2	OP+1	ii	1	OP+1	dd	1	OP+1	hh	1	OP+1	ff	1	OP+1	A6	1
3				00dd	(00dd)		ll		1			FFFF	—		1
	ff		1					OP+2		ll	1			OP+2	
4							hhll	(hhll)	1	X+ff	(X+ff)	1	FFFF	—	1
5													Y+ff	(Y+ff)	1

OP - Prevzem ukaza
- branje ukaza iz pomnilnika

ff - odmik za indeksno naslavljanje

ii - takojšnji operand

dd - pom. naslov za neposredno pom. nas.

FFFF --- - v tem ciklu na vodilu ni prenosa

hhll - pom. naslov za razširjeno neposredno pom. naslavljanje

LDA

Načini naslavljanja, Strojni ukazi, Izvajanje po ciklih:

Cikel	LDAA (IMM)			LDAA (DIR)			LDAA (EXT)			LDAA (IND,X)			LDAA (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	86	1	OP	96	1	OP	B6	1	OP	A6	1	OP	18	1
2	OP+1	ii	1	OP+1	dd	1	OP+1	hh	1	OP+1	ff	1	OP+1	A6	1
3				00dd	(00dd)	1	OP+2	ll	1	FFFF	—	1	OP+2	ff	1
4							hhll	(hhll)	1	X+ff	(X+ff)	1	FFFF	—	1
5													Y+ff	(Y+ff)	1

86=1000 0110 96=1001 0110 B6=1011 0110 A6=1010 0110 A6=1010 0110

Cikel	LDAB (IMM)			LDAB (DIR)			LDAB (EXT)			LDAB (IND,X)			LDAB (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	C6	1	OP	D6	1	OP	F6	1	OP	E6	1	OP	18	1
2	OP+1	ii	1	OP+1	dd	1	OP+1	hh	1	OP+1	ff	1	OP+1	E6	1
3				00dd	(00dd)	1	OP+2	ll	1	FFFF	—	1	OP+2		1
4							hhll	(hhll)	1	X+ff	(X+ff)	1	FFFF	—	1
5													Y+ff	(Y+ff)	1

C6=1100 0110 D6=1101 0110 F6=1111 0110 E6=1110 0110 E6=1110 0110

Ukazi za seštevanje

- **Skupina ukazov ADD prišteje vrednost registrov, takojšnjih operandov, ali pomnilniških lokacij k akumulatorju.**
  - ABA - prištej ak. B k ak. A
  - ABX - prištej ak. B k indeksnem reg. X
  - ABY - prištej ak. B k indeksnem reg. Y
  - ADDA #$13 - prištej $13 k ak. A
  - ADDB $64 - prištej mem[$64] k ak. B

Seštevanje s prenosom, Odštevanje

- **ADC prišteje dve vrednosti k akumulatorju**
  - Podan operand in bit C (prenos)
  - Omogoča seštevanje števil, daljših od 16 bitov
  - ADCA #72 - prišteje 72 + C bit k ak. A
  - ADCB $0112 - prišteje mem[$0112] + C k ak. B

- **Odštevanje je podobno seštevanju**
  - SBA, SUBA, SUBB, SUBD, SBCA, SBCB
- **Pri seštevnaju/odštevanju se ustrezno postavijo zastavice H*, N, Z, V, C**

  ***na H vplivajo samo ukazi za seštevanje**
  - ADDD 10,Y - prištej mem[10+Y],[10+Y+1] k ak. D
    - Če Y = 30 in mem[40] = $12, mem[41]=$A2, potem se k D prišteje $12A2
- **Zastavica C je uporabna za seštevanje "daljših" števil**

Pisanje v pomnilnik

- **Pisanju v pomnilnik pravimo tudi *shranjevanje* podatkov**
- **Za to uporabljamo skupino ukazov STORE**
  - STAA $12,Y - shrani ak. A v mem[$12+Y]
  - STAB $3412 - shrani ak. B v mem[$3412]
  - STD $102 - shrani ak. D v mem[$102],[$103]
  - STX $3FF2 - shrani X v mem[$3FF2],[$3FF3]
  - STY 18,X - shrani Y v mem[18+X],[18+X+1]
  - STS $44 - shrani SP v mem[$44],[$45]
- **Ukazi STORE vplivajo na zastavice enako kot ukazi LOAD:**

S	X	H	I	N	Z	V	C
-	-	-	-	⇳	⇳	0	-

Seštevanje dolgih števil

- **Kako seštevamo števila, ki so daljša od 8 bitov?**

```
   11
  $22C7
+ $3159
  -----
  $5420
```

Želeni rezultat

```
        ORG     $2100
RESULT  RMB     2
        LDAA    #$C7
        ADDA    #$59    ;add lower byte
        STAA    RESULT+1
        LDAA    #$22
ADCA    ADDA    #$31    ;add upper byte
        STAA    RESULT
        END
```

ADCA - Add with carry to A

```
    1
  $22C7
+ $3159
  -----
  $5320
```

Rezultat

Program deluje le, če ukaza STAA and LDAA ne spremenita zastavice prenosa C! Preveri !!

•Odštevanje je podobno seštevanju, le da namesto prenosa upoštevamo **borrow**. *(zastavica C pri odštevanju predstavlja borrow)*

Ukazi Exchange/Transfer

- **Ukazi za zamenjavo vrednosti dveh registrov**
  - **`XGDX`** - Exchange (swap) X <=> D
  - **`XGDY`** - Exchange (swap) Y <=> D
- **Prepisovanje vrednosti enega registra v drugega**
  - **`TAB`** - Transfer (kopiraj) A => B (ni zamenjava vrednosti!)
  - **`TBA`** - B => A
  - **`TSX, TSY`** - Transfer (stack pointer+1) => X (ali Y)
  - **`TXS, TYS`** - Transfer [X (ali Y)-1] => SP
  - **`TPA`** - Transfer CCR (reg. pogojnih kod) => ak. A
  - **`TAP`** - Transfer ak. A => CCR

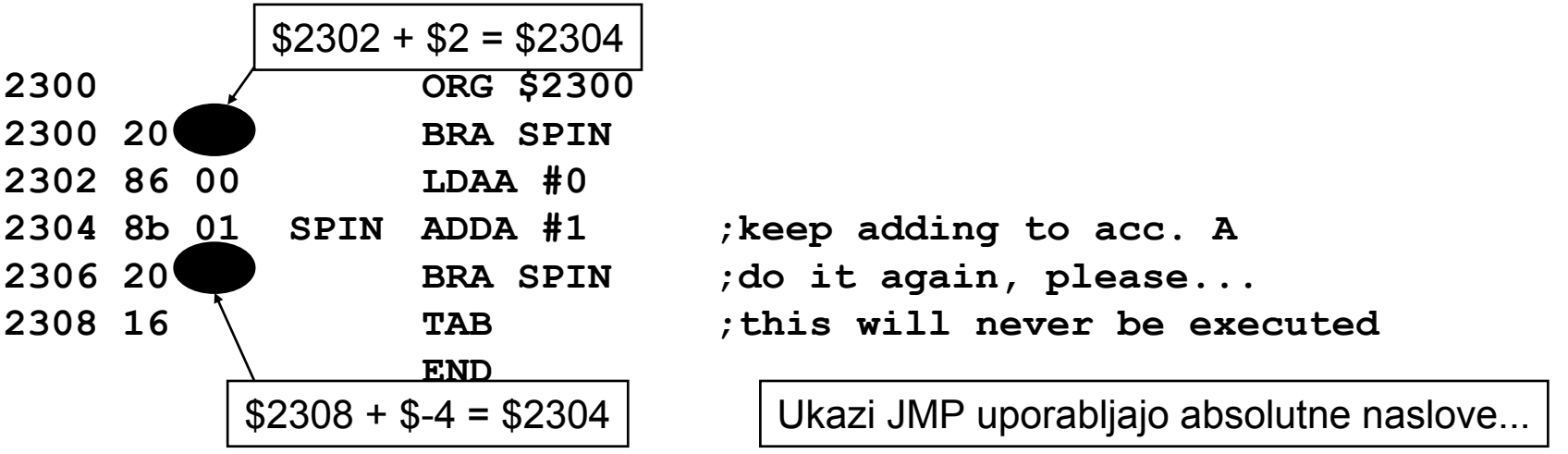
Zanke in Vejitve

- **Zanke naredimo z uporabo ukazov tipa GoTo**
  - Pogojni skoki (samo kratki - BXX)
  - Brezpogojne skoke predstavljata ukaza **`BRA`** (kratki) in **`JMP`** (dolgi)

```
        ORG    $2300
        BRA    SPIN   ;skip next instruction
        LDAA   #0     ;too bad we're not executing this...
SPIN    ADDA   #1     ;keep adding to acc. A
        BRA    SPIN   ;do it again, please...
        END
```

Naslavljanje pri ukazih Bxx

- **Vejitve uporabljajo relativno naslavljanje**
  - Skočni naslov pri ukazu Bxx je
    - Določen kot odmik glede na NASLEDNJI ukaz. NASLOV=PC+ODMIK
    - Odmik je 8-bitno predznačeno št. v dvojiškem komplementu



Zastavice (Condition Codes)

- **Zastavice postavi večina aritmetičnih ukazov glede na rezultat operacije.**
  - `ADDA $2020` ; if carry is produced, Carry CC is set
- **Zastavice so veljavne po končanem ukazu, ki jih je postavil**
  - Veljavne ostanejo, dokler jih ne spremeni kakšen drug ukaz

```
ADDA    $2020
BCC     FOO
```

Postavi zastavice C, Z, N, V, in H

Pogojni skok glede na zastavice, ki jih je postavil ADD (Branch Carry Clear)

- **Zastavice**
  - C - Prenos (za aritmetične ukaze)
  - Z - Zero (postavljen, če je rezultat operacije nič)
  - N - Negative (postavljen, če je rezultat operacije negativen)
  - V - Preliv (za aritmetične ukaze)
  - H - Half Carry prenos od bita 3 do 4

Ukazi Bxx

BCC	Carry Clear	$\overline{C}$
BCS	Carry Set	C
BEQ	Equal	Z
BNE	Not Equal	$\overline{Z}$
BMI	Minus	N
BPL	Plus	$\overline{N}$
BVC	Overflow clear	$\overline{V}$
BVS	Overflow set	V
BHI	Higher	$C \cdot \overline{Z}$
BHS	Higher or Same	$\overline{C}$
BLO	Lower	C
BLS	Lower or Same	C + Z
BGE	Greater or Equal	$N \cdot V + \overline{N} \cdot \overline{V}$
BGT	Greater	$(N \cdot V \cdot + \overline{N} \cdot \overline{V}) \cdot \overline{Z}$
BLE	Less than or Equal	$Z + \overline{N} \cdot V + N \cdot \overline{V}$
BLT	Less than	$N \cdot \overline{V} + \overline{N} \cdot V$

Ukazi so smiselni, če je zastavice postavil ukaz, ki odšteje $N_1 - N_2$

Za aritmetiko z nepredznačenimi števili

Za aritmetiko v dvojiškem komplementu

Ukaz za primerjanje (Compare)

- **Ukaz compare se uporablja za postavitev zastavic**
  - `CMPA $2030`

Izvede se odštevanje **A - mem[$2030],** rezultat se zavrže, zastavice pa se postavijo enako, kot če bi se izvedel ukaz SUBA

- **CMP pripravi zastavice za pogojni skok, ki sledi**

```
LOOP    ADDA    #1
        CMPA    #32
        BNE     LOOP
```

Ponavljaj, dokler ni izpolnjen pogoj
A == 32

Preveri ali Z==0
Branch Not Equal

Različice: **`CMPA, CMPB, CBA, CPD, CPX, CPY`**

Uporaba CMP in Bxx

Izračuna A - <K>

Izvedi ukaz **CMPA <K>** . **Če mu sledi ukaz** Bxx, to pomeni:

Ukaz	Skoči če:
BEQ	A = K
BNE	A ≠ K
BLT	A < K
BLE	A ≤ K
BGT	A > K
BGE	A ≥ K

Za **predznačena števila**.

Ukaz	Skoči če:
BEQ	A = K
BNE	A ≠ K
BLO	A < K
BLS	A ≤ K
BHI	A > K
BHS	A ≥ K

Za **nepredznačena števila**.

Ukazi Bxx se običajno uporabljajo za ukazom CMP.