Styrenheten – styrsignalsekvenser minnesoperander

Dagens föreläsning behandlar:

Kompendiet kapitel 8.5
Arbetsboken kapitel 14

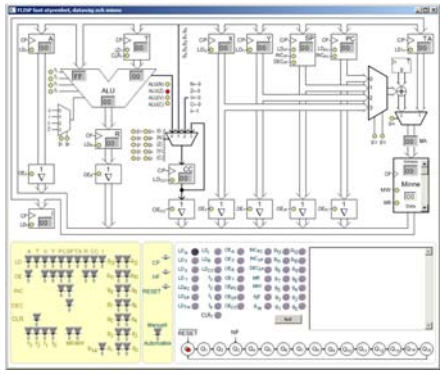
Ur innehållet:

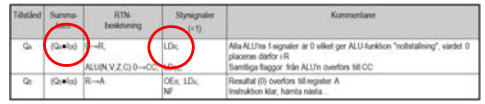
- Konstruktion och implementering av styrsignalsekvenser för maskininstruktioner
- Läsning, skrivning av minnesoperand (”load/store”)
- Direkt modifiering av minnesoperand (”read/modify/write”)
- Indexerade adresseringssätt

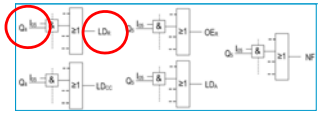
FLIS-processorns automatiska styrenhet

Den automatiska styrenheten genererar sekvenser med styrsignalinformation.
En specifik instruktion implementeras med en unik styrsignalsekvens.

EXEMPEL; **CLRA** (Clear register A)





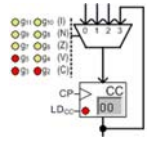


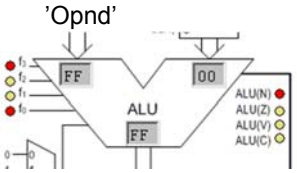
```
CLRA (05)
# MergeState LDR=(I05*Q4)
# Flaggsättning
; ALU(N)->CC(N); ALU(Z)->CC(Z);ALU(V)->CC(V); ALU(C)->CC(C);
# MergeState LDCC=(I05*Q4)
# MergeState OER=(I05*Q5)
# MergeState LDA=(I05*Q5)
# MergeState NF=(I05*Q5)
```

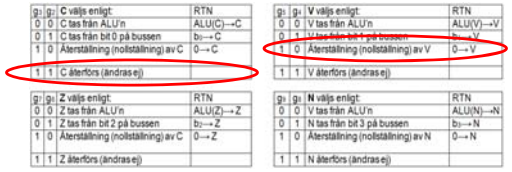

Utförandefasen av instruktionen LDY **LDY** flaggsättning

Instruktionen har fem olika adresseringssätt (varianter) med unika operationskoder.
Var och en implementeras för sig men har stora likheter.
Låt ’Opnd’ beteckna det värde som ska laddas i Y

Instruktion LD Variant		Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
LDY	#Data	Immediate	91	2	2	Data→Y	Δ	Δ	0	-
LDY	Adr	Absolute	A1	2	3	M(Adr)→Y				
LDY	n,SP	Indexed	B1	2	3	M(n+SP)→Y				
LDY	n,X	Indexed	C1	2	3	M(n+X)→Y				
LDY	n,Y	Indexed	D1	2	3	M(n+Y)→Y				







D + C_{in} → U,
ger korrekt flaggsättning för Z och N

ALU(N,V) →CC;
0→CC(V);
CC(C) →CC(C);

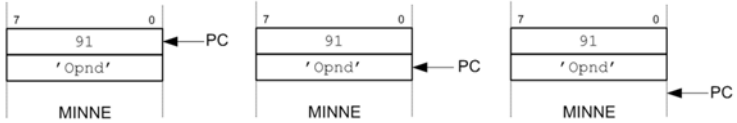
f_3; f_1;
g_5;
g_3; g_2;
LD_{CC};

Flaggsättning sker..

Utförandefasen av instruktionen LDY **LDY** #Data

Instruktion LD Variant		Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
LDY	#Data	Immediate	91	2	2	Data→Y	Δ	Δ	0	-
LDY	Adr	Absolute	A1	2	3	M(Adr)→Y				
LDY	n,SP	Indexed	B1	2	3	M(n+SP)→Y				
LDY	n,X	Indexed	C1	2	3	M(n+X)→Y				
LDY	n,Y	Indexed	D1	2	3	M(n+Y)→Y				

Före ’FETCH’ Före ’EXECUTE’ Efter ’EXECUTE’



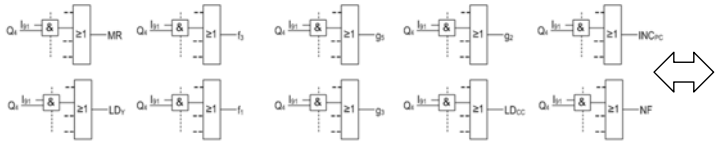
M(PC)→Y;
PC+1→PC

⇔ MR; LD_Y;INC_{PC};

Utförandefasen av instruktionen LDY

Sammanfattning: **LDY** #Data

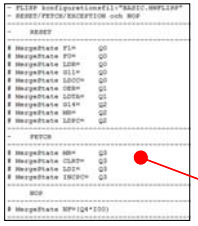
Tillstånd	Summa-term	RTN-beskrivning	Styrsignaler (=1)	Kommentarer
Q_4	$(Q_4 \bullet I_{91})$	M(PC)→Y; ALU(N,V) →CC; 0→CC(V); CC(C) →CC(C); PC+1→PC;	MR; LD_Y; f_3; f_1; g_5; g_3; g_2; LD_{CC}; INC_{PC}; NF	'Opnd' läses direkt från minnet till register Y Flaggsättning sker.. PC uppdateras förbi 'Opnd' Instruktion klar, hämta nästa...




```
; Implementering av LDY #Data
; Data->Y
# MergeState LDY=(I91*Q4)
# MergeState INCPC=(I91*Q4)
# MergeState MR=(I91*Q4)

; Flaggsättning
# MergeState F3=(I91*Q4)
# MergeState F0=(I91*Q4)
# MergeState G5=(I91*Q4)
# MergeState G3=(I91*Q4)
# MergeState G2=(I91*Q4)
# MergeState LDCC=(I91*Q4)

# MergeState NF=(I91*Q4)
```


Utförandefasen av instruktionen LDY

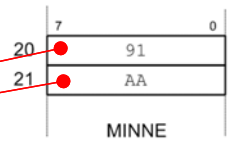
Test av: **LDY** #Data

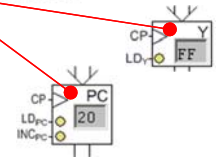


```
; Implementering av LDY #Data
; Data->Y
# MergeState LDY=(I91*Q4)
# MergeState INCPC=(I91*Q4)
# MergeState MR=(I91*Q4)

; Flaggsättning
# MergeState F3=(I91*Q4)
# MergeState F0=(I91*Q4)
# MergeState G5=(I91*Q4)
# MergeState G3=(I91*Q4)
# MergeState G2=(I91*Q4)
# MergeState LDCC=(I91*Q4)

# MergeState NF=(I91*Q4)
```

```
- "Test_LDY_imm.HWFLISP"
- Konfigurationsfil för test av LDY #Data
-----------------------------------------

; Ta bort alla gamla styrsignalsekvenser
# ClearAllStates
; Läs in nya styrsignalsekvenser
# load "basic.hwflisp"
# load "loads.hwflisp"

; Testprogrammet
# setMemory 20=91
# setMemory 21=AA

; Sätt initiala registervärden
# setRegister Y=FF
# setRegister PC=20
```

EXEMPEL:
LDY #AA_{16}





Utförandefasen av instruktionen LDY

LDY Adr

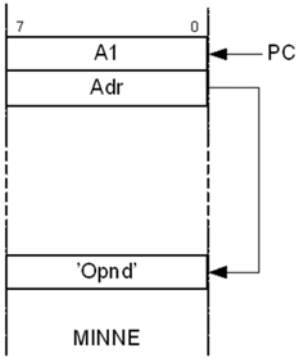
Instruktion LD Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
LDY #Data	Immediate	91	2	2	Data→Y	Δ	Δ	0	-
LDY Adr	Absolute	A1	2	3	M(Adr)→Y				
LDY n,SP	Indexed	B1	2	3	M(n+SP)→Y				
LDY n,X	Indexed	C1	2	3	M(n+X)→Y				
LDY n,Y	Indexed	D1	2	3	M(n+Y)→Y				

Detta adresseringssätt kräver **två läsningar** i minnet, först läses adressen till operanden (Adr) och först därefter kan operanden 'Opnd' läsas.

Under Q4:
M(PC)→TA;
PC+1→PC;

Under Q5:
M(TA)→Y;

Före 'FETCH'



Tillstånd	Summa-term	RTN-beskrivning	Styrsignaler (=1)	Kommentarer
Q_4	$(Q_4 \bullet I_{A1})$	M(PC)→TA; PC+1→PC;	MR; LD_{TA}; INC_{PC};	'Adr' läses från minnet till TA PC uppdateras förbi 'Adr'
Q_5	$(Q_5 \bullet I_{A1})$	M(TA)→Y; ALU(N,V) →CC; 0→CC(V); CC(C) →CC(C);	MR; LD_Y; f_3; f_1; g_5; g_3; g_2; LD_{CC}; NF	'Opnd' läses nu från minnet till register Y Flaggsättning sker.. Instruktion klar, hämta nästa...

Sammanfattning:
LDY Adr

```
; Implementering av LDY Adr
; Adr->TA; PC+1->PC;
# MergeState LDTA=(IA1*Q4)
# MergeState INCPC=(IA1*Q4)
# MergeState MR=(IA1*Q4)
; (Adr)->Y
# MergeState G14=(IA1*Q5)
# MergeState LDY=(IA1*Q5)
# MergeState MR=(IA1*Q5)
; Flaggsättning
# MergeState F3=(IA1*Q5)
# MergeState F0=(IA1*Q5)
# MergeState G5=(IA1*Q5)
# MergeState G3=(IA1*Q5)
# MergeState G2=(IA1*Q5)
# MergeState LDCC=(IA1*Q5)

# MergeState NF=(IA1*Q5)
```

Jämför med
`LDY #Data`

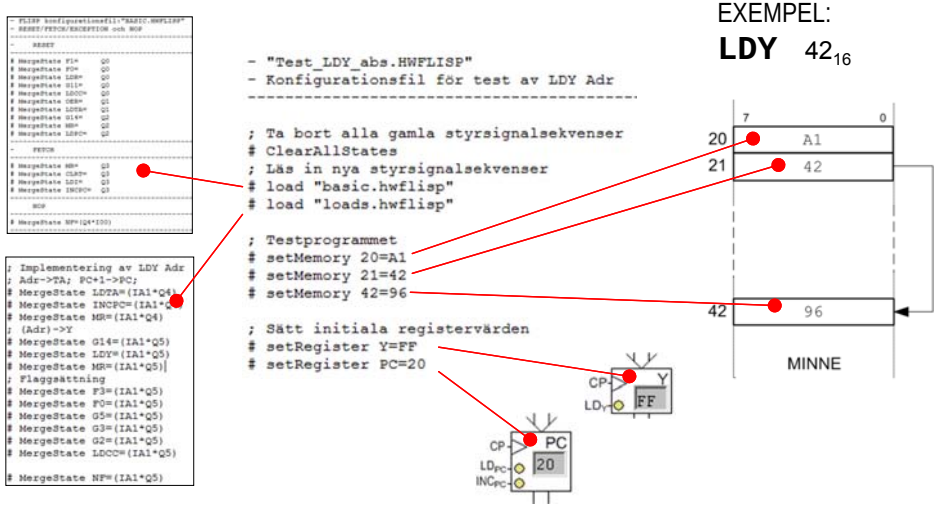
```
; Implementering av LDY #Data
; Data->Y
# MergeState LDY=(I91*Q4)
# MergeState INCPC=(I91*Q4)
# MergeState MR=(I91*Q4)

; Flaggsättning
# MergeState F3=(I91*Q4)
# MergeState F0=(I91*Q4)
# MergeState G5=(I91*Q4)
# MergeState G3=(I91*Q4)
# MergeState G2=(I91*Q4)
# MergeState LDCC=(I91*Q4)

# MergeState NF=(I91*Q4)
```

Utförandefasen av instruktionen LDY

Test av: **LDY** Adr

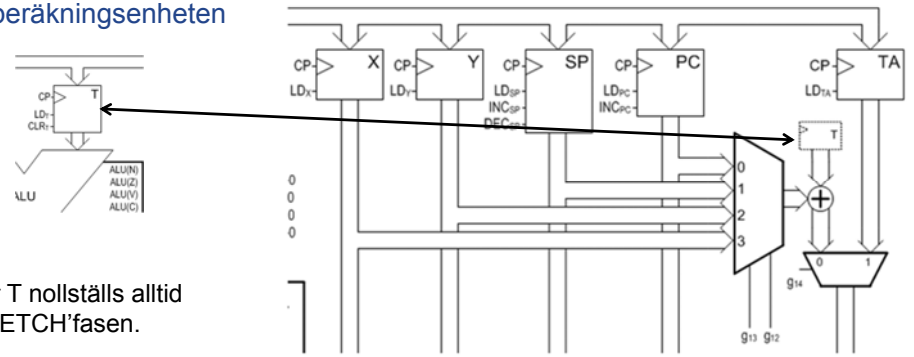
EXEMPEL:

LDY 42_{16}



Indexerad adressering –

adressberäkningsenheten



Register T nollställs alltid under 'FETCH'fasen.

Under Q4:

M(PC)→T;
PC+1→PC;

Under Q5:

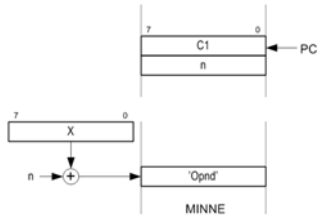
M(T+X)→Y;

g_{14}	g_{13}	g_{12}	Register till adressbuss:	RTN
0	0	0	Register PC	M(PC)
0	0	1	Register SP ("bas") och register T ("offset")	M(T+SP)
0	1	0	Register Y ("bas") och register T ("offset")	M(T+Y)
0	1	1	Register X ("bas") och register T ("offset")	M(T+X)
1	0	0	Adressberäkningsregister, ingen offset	M(TA)
1	0	1	Adressberäkningsregister, ingen offset	M(TA)
1	1	0	Adressberäkningsregister, ingen offset	M(TA)
1	1	1	Adressberäkningsregister, ingen offset	M(TA)

Utförandefasen av instruktionen LDY

LDY n,X

Instruktion LD Variant		Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
LDY	#Data	Immediate	91	2	2	Data→Y	Δ	Δ	0	-
LDY	Adr	Absolute	A1	2	3	M(Adr)→Y				
LDY	n,SP	Indexed	B1	2	3	M(n+SP)→Y				
LDY	n,X	Indexed	C1	2	3	M(n+X)→Y				
LDY	n,Y	Indexed	D1	2	3	M(n+Y)→Y				



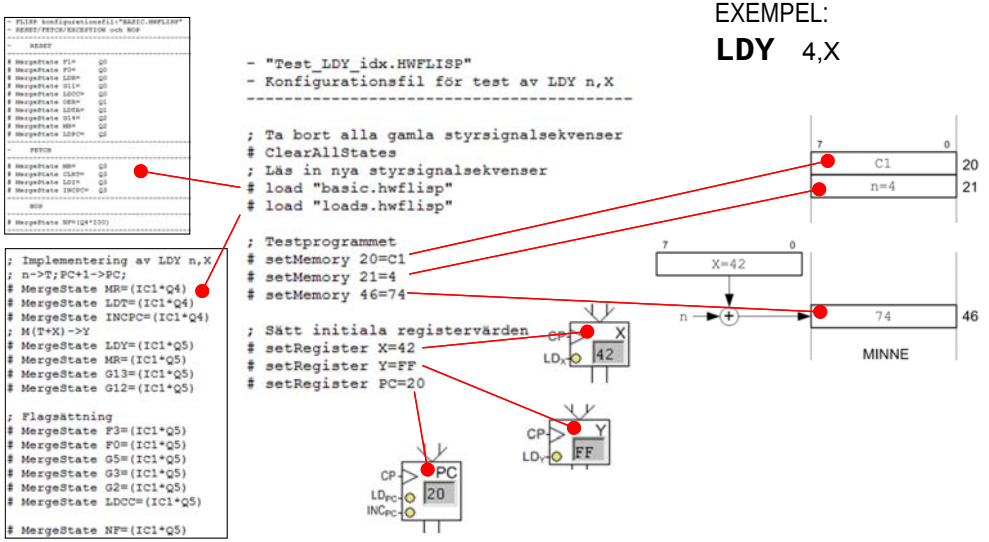
Tillstånd	Summa-term	RTN-beskrivning	Styrsignaler (=1)	Kommentarer
Q_4	$(Q_4 \bullet I_{C1})$	M(PC)→T; PC+1→PC;	MR; LD_T; INC_{PC};	'n' läses från minnet till T PC uppdateras förbi 'n'
Q_5	$(Q_5 \bullet I_{C1})$	M(T+X)→Y; ALU(N,V) →CC; 0→CC(V); CC(C) →CC(C);	MR; g_{13}; g_{12}; LD_Y; f_3; f_1; g_5; g_3; g_2; LD_{CC}; NF	'Opnd' läses nu från minnet till register Y via register X Flaggsättning sker.. Instruktion klar, hämta nästa...

Utförandefasen av instruktionen LDY

Test av: **LDY** n,X

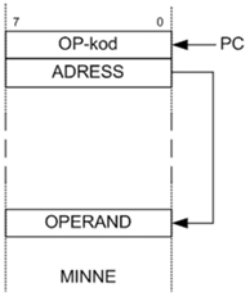
EXEMPEL:

LDY 4,X



Utförandefasen av instruktionen STY

Instruktion ST Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
STY Adr	Absolute	31	2	3	Y → M(Adr)	-	-	-	-
STY n,SP	Indexed	41	2	3	Y → M(n+SP)				



STY Adr

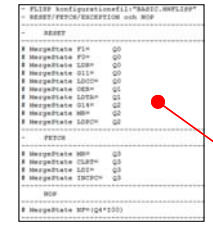
I tillstånd Q_4 läses adressen och placeras i register TA.
I tillstånd Q_5 skrivs innehållet i register Y till adressen.

Tillstånd	Summa-term	RTN-beskrivning	Styrsignaler (=1)	Kommentarer
Q_4	$(Q_4 \bullet I_{31})$	M(PC)→TA; PC+1→PC;	MR; LD_{TA}; INC_{PC};	ADRESS läses från minnet till TA PC uppdateras förbi ADRESS
Q_5	$(Q_5 \bullet I_{31})$	Y→ M(TA);	MW; g_{14}; OE_Y; NF	register Y skrivs till minnet Instruktion klar, hämta nästa...

Utförandefasen av instruktionen STY

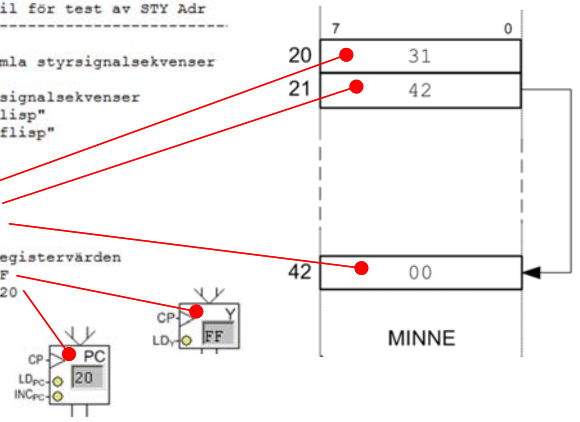
Test av: **STY** Adr

EXEMPEL:
STY 42_{16}



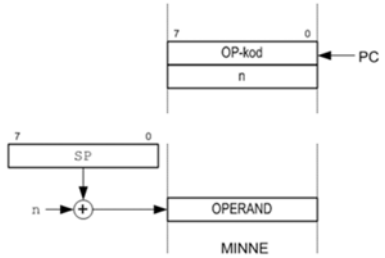
```
; Implementering av STY Adr
; Adr->TA; PC+1->PC;
# MergeState LDTA=(I31*Q4)
# MergeState INCPC=(I31*Q4)
# MergeState MR=(I31*Q4)
; Y->(Adr)
# MergeState G14=(I31*Q5)
# MergeState OEY=(I31*Q5)
# MergeState MW=(I31*Q5)
# MergeState NF=(I31*Q5)
```

```
- "Test_STY_abs.HWFLISP"
- Konfigurationsfil för test av STY Adr
-----------------------------------------

; Ta bort alla gamla styrsignalsekvenser
# ClearAllStates
; Läs in nya styrsignalsekvenser
# load "basic.hwflisp"
# load "stores.hwflisp"

; Testprogrammet
# setMemory 20=31
# setMemory 21=42
# setMemory 42=00

; Sätt initiala registervärden
# setRegister Y=FF
# setRegister PC=20
```



Utförandefasen av instruktionen STY

Instruktion ST Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
STY Adr	Absolute	31	2	3	Y → M(Adr)	-	-	-	-
STY n,SP	Indexed	41	2	3	Y → M(n+SP)				



STY n,SP

I tillstånd Q_4 läses offset 'n' och placeras i register T.
I tillstånd Q_5 skrivs innehållet i register Y till adressen M(T+SP).

Tillstånd	Summa-term	RTN-beskrivning	Styrsignaler (=1)	Kommentarer
Q_4	$(Q_4 \bullet I_{41})$	M(PC)→T; PC+1→PC;	MR; LD_T; INC_{PC};	'n' läses från minnet till T PC uppdateras förbi 'n'
Q_5	$(Q_5 \bullet I_{41})$	Y→ M(T+SP);	MW; g_{12}; OE_Y; NF	register Y skrivs till minnet Instruktion klar, hämta nästa...

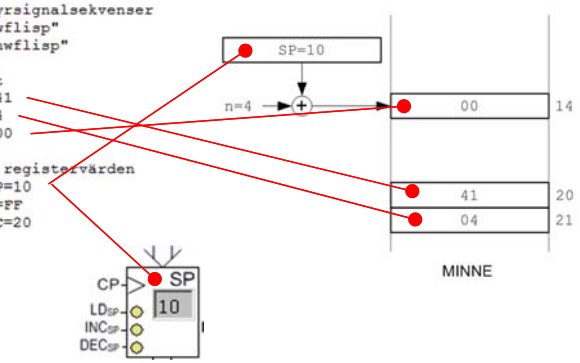
Utförandefasen av instruktionen STY

Test av: **STY** n,SP

EXEMPEL:
STY **4, SP**

```
; Implementering av STY n,SP
; n->T;PC+1->PC;
# MergeState MR=(I41*Q4)
# MergeState INCPC=(I41*Q4)
# MergeState LDT=(I41*Q4)
; Y->M(T+SP)
# MergeState G12=(I41*Q5)
# MergeState OEY=(I41*Q5)
# MergeState MW=(I41*Q5)
# MergeState NF=(I41*Q5)
```

```
- "Test_STY_idx.HWFLISP"
- Konfigurationsfil för test av STY n,SP
-----------------------------------------

; Ta bort alla gamla styrsignalsekvenser
# ClearAllStates
; Läs in nya styrsignalsekvenser
# load "basic.hwflisp"
# load "stores.hwflisp"

; Testprogrammet
# setMemory 20=41
# setMemory 21=4
# setMemory 14=00

; Sätt initiala registervärden
# setRegister SP=10
# setRegister Y=FF
# setRegister PC=20
```



”read/modify/write”

Utförandefasen delas in i tre steg:

1. Operandens adress bestäms, detta kan innebära att adressen placeras i register TA, eller, om indexerad adressering används att en offset placeras i register T.
2. Operationen utförs i ALU:n och resultatet sparas i register R, under detta steg sker också flaggsättningen.
3. Resultatet i register R skrivs tillbaks till operandens adress.

Observera att de register som bildar operandens adress inte får ändras mellan 1. och 3.

Utförandefasen av NEG Adr

Instruktion NEG Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
NEG Adr	Absolute	36	2	4	~ M(Adr) → M(Adr)	Δ	Δ	Δ	Δ

Flaggor
N: Ettställs om resultatets teckenbit (bit 7) får värdet 1.
Z: Ettställs om samtliga åtta bitar i resultatet blir noll, dvs om det gamla värdet är noll.
V: Ettställs om 2-komplementoverflow uppstår.
C: Ettställs om det gamla värdet ≠ 0.

1. M(PC)→TA; PC+1→PC

Operationen utförs och flaggorna sätts:

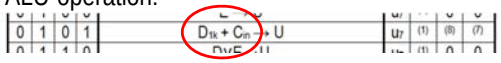
2. $M(TA)_{1k} + 1 \rightarrow R$; ALU(CC)→CC

Resultatet skrivs tillbaks till operandens adress i minnet

3. R→ M(TA);

ALU:n sätter flaggorna enligt instruktionens specifikation.

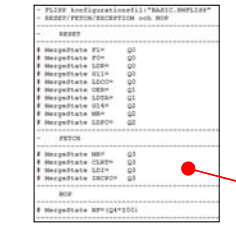
ALU-operation:



Tillstånd	Summa-term	RTN-beskrivning	Styrsignaler (=1)	Kommentarer
Q_4	(Q_4•I_{36})	M(PC)→TA; PC+1→PC;	MR; LD_{TA}; INC_{PC};	'Adr' läses från minnet till TA PC uppdateras förbi 'Adr'
Q_5	(Q_5•I_{36})	$M(TA)_{1k}$ + 1→R; ALU(CC)→CC	MR;g_{14};f_2;f_0;g_0; LD_R; LD_{CC}	Läsning och tvåkomplementering flaggsättning
Q_6	(Q_6•I_{36})	R→ M(TA);	MW; g_{14}; OE_R; NF	register R skrivs till minnet Instruktion klar, hämta nästa...

Utförandefasen av instruktionen NEG Adr

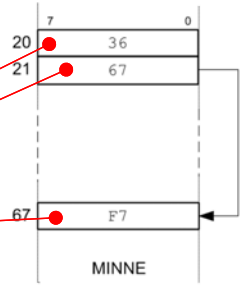
Test av: **NEG Adr**



```
    NEG  Adr
; Adr->TA; PC+1->PC;
# MergeState LDTA=     (I36*Q4)
# MergeState INCPC=    (I36*Q4)
# MergeState MR=       (I36*Q4)
; M(TA)~1->R, Cin=1
# MergeState MR=       (I36*Q5)
# MergeState G14=      (I36*Q5)
# MergeState F2=       (I36*Q5)
# MergeState F0=       (I36*Q5)
# MergeState G0=       (I36*Q5)
# MergeState LDR=      (I36*Q5)
; Flaggsättning
; ALU(CC)->CC
# MergeState LDCC=     (I36*Q5)
; R->M(TA)
# MergeState OER=      (I36*Q6)
# MergeState MW=       (I36*Q6)
# MergeState G14=      (I36*Q6)
# MergeState NF=       (I36*Q6)
```

Test av: **NEG 67_{16}**

```
- "Test_NEG_abs.HWFLISP"
- Konfigurationsfil för test av NEG Adr
-----------------------------------------

; Ta bort alla gamla styrsignalsekvenser
# ClearAllStates
; Läs in nya styrsignalsekvenser
# load "basic.hwflisp"
# load "unary.hwflisp"

; Testprogrammet
# setMemory 20=36
# setMemory 21=67
# setMemory 67=F7

; Sätt initiala registervärden
# setRegister PC=20
```



Anm: $F7_{16}=(-9)_{10}$