Mikroprogrammering I

Att bygga en CPU

Olle Roos-datorn (fö2+3)
Björn Lindskog-datorn (lab1)

- Variabel exekveringstid
- Variabelt format
- Inget överlapp
- Central styrenhet, som är mikroprogrammerad
- Flera adresseringsmoder/instruktion
- 1 ackumulator
- Nästan alla instruktioner har operand i minnet:
  LDA A ; A=M(A)
  ADDA A ; A=A+M(A)

Pipelinad dator (fö4,lab2)

- Alla instruktioner tar 5 CK
- Alla instruktioner har samma format
- Pipelining/överlapp ger 1 färdig instruktion/CK
- Flera avkodare (inget µprog)
- 1 a-mod/instruktion
- 32 register
- Endast LD/ST har operand i minnet:
  LD Rd,(Ra) ; Rd=M(Ra)
  ADD Rd,Ra,Rb ; Rd=Ra+Rb

2

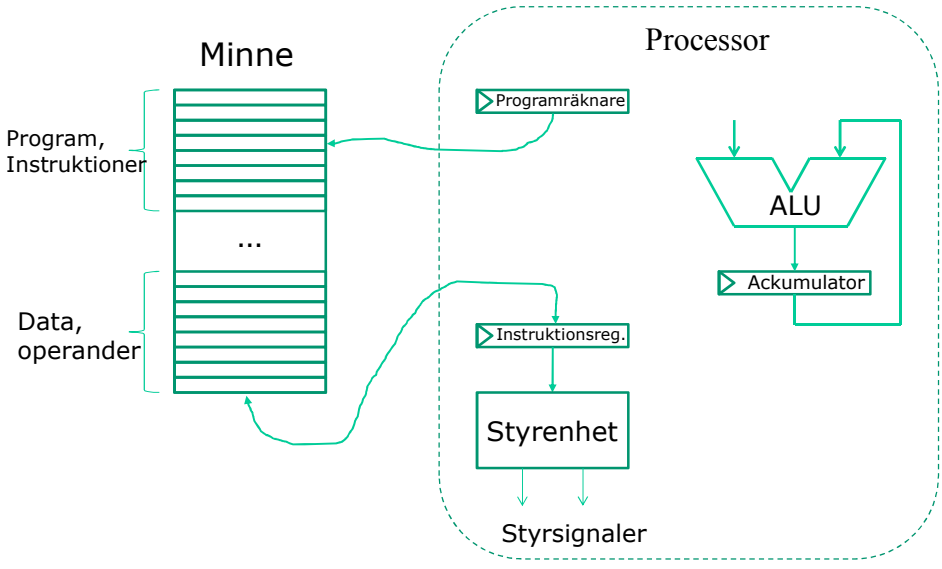
Att bygga en CPU

- Typisk CISC
- Programmering på 2 nivåer asm och mikro
- Enkel controller:
  - garageportsöppnare
  - del av dator
- + Man kan göra avancerade instruktioner: **sortera**
- **-** Många CK/instr blir det …

- Typisk RISC
- Programmering på 1 nivå: asm
- Enkel CPU
  - enkel mobil
- Finns bara enkla instr.
- Snabb

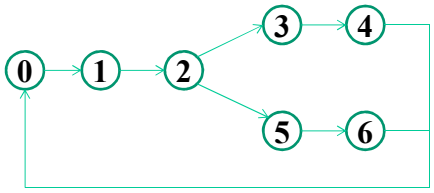
3

Ritning 1



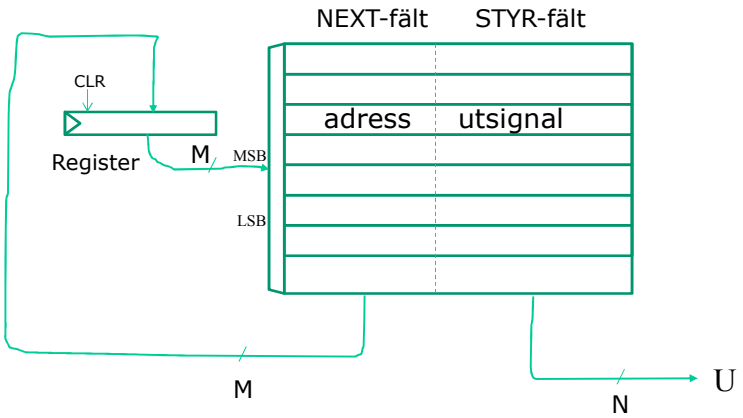
Mikroprogrammering

- Vi ska bygga en liten dator med enkla komponenter
- Styrenheten (sekvensnät) visar sig vara svårast. Hur gör man för att konstruera ett SN med 100 tillstånd?
- mikroprogrammering är en vidareutveckling (och faktiskt begränsning) av tekniken att bygga sekvensnät med ROM
- idé: byt tillståndsvipporna mot en universalräknare
- Moore, få hopp



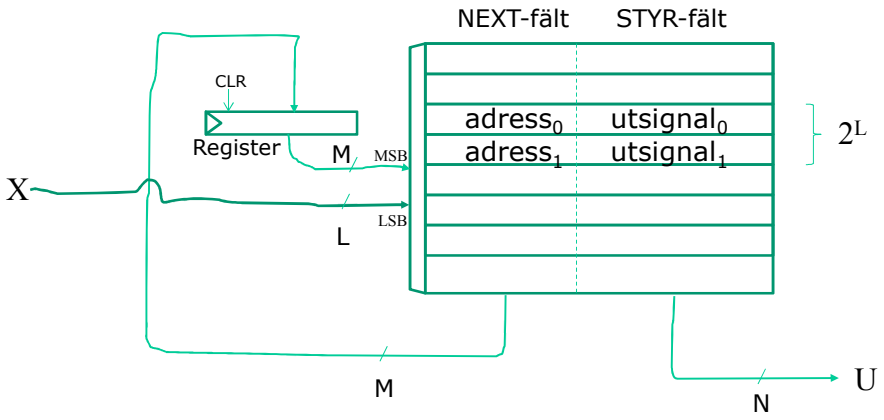
Ritning2

Autonom styrenhet med ROM/Register



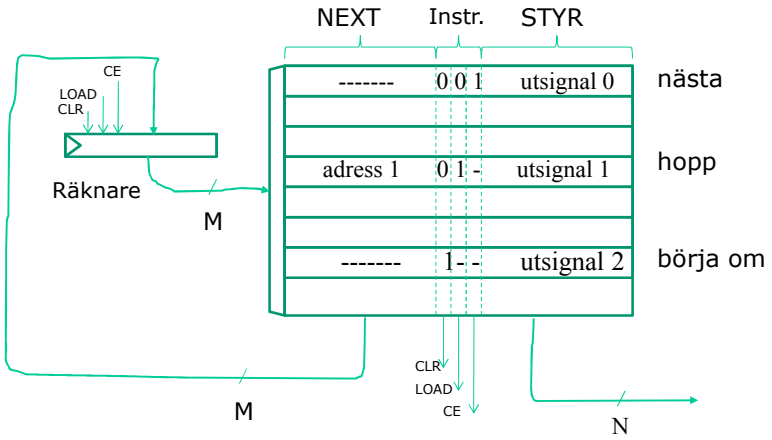
Variant2

Styrenhet med ROM/Register

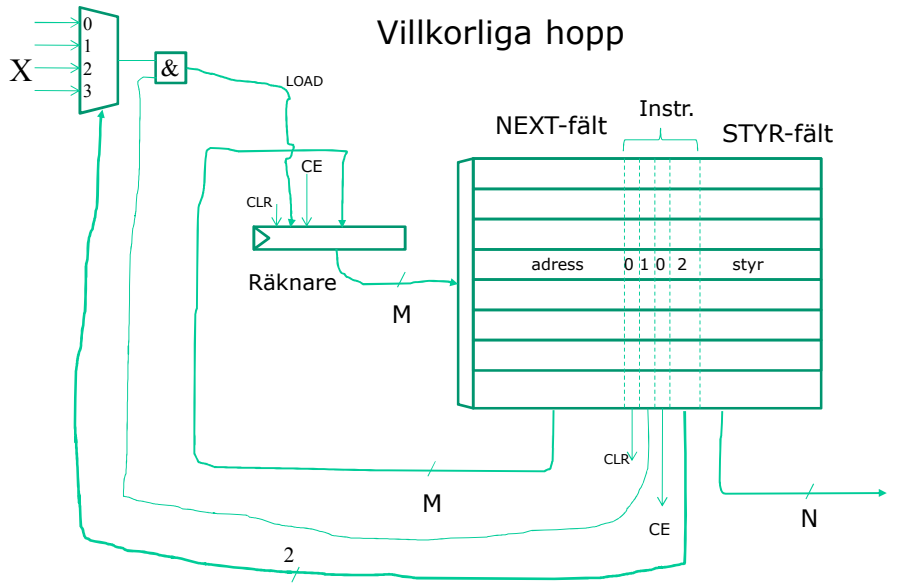


Ritning 3

Autonom styrenhet med ROM/Räknare

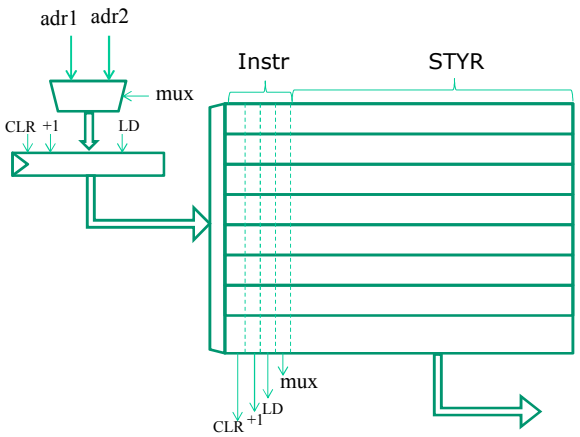


Ritning 4



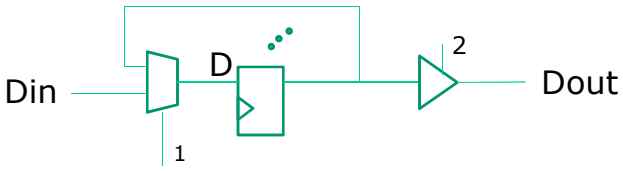
En variant

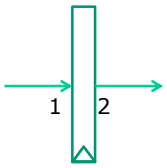
Autonom styrenhet utan adressfält

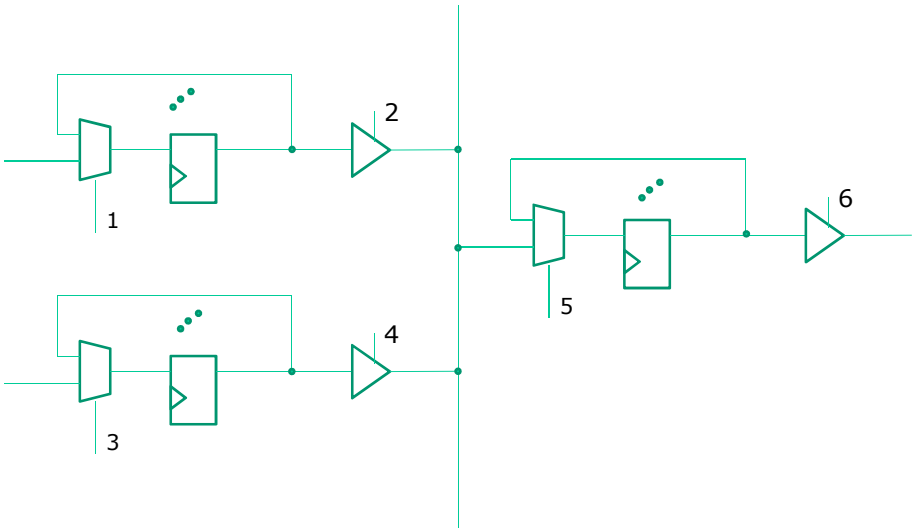


Ritning 5

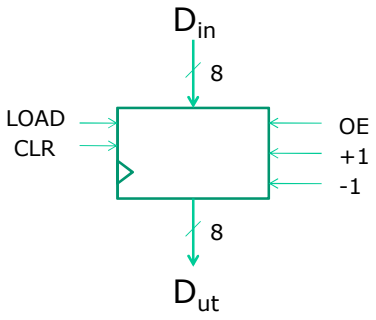
Register



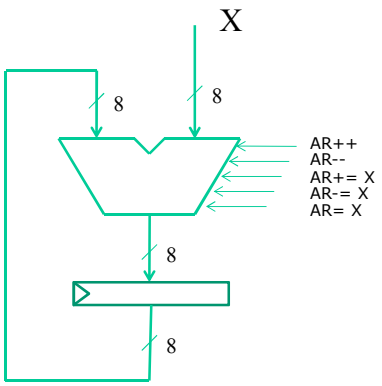




Ritning 6 och 7

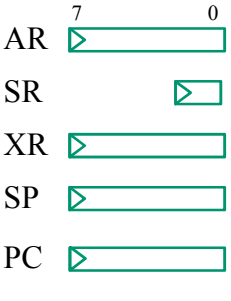


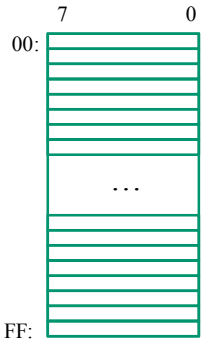
universalräknare



ALU/ackumulator

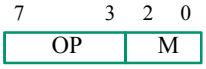
Ritning 8 - Programmerarmodell





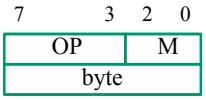
Endast 2 flaggor: Z,N

Ritning 9 - instruktionsformat



Opkod
32 instruktioner

Märkfält
8 adresseringsmoder



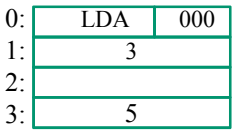
Adress/data

Adresseringsmoder

M	Mod	Exempel	EA
000	Absolut	LDA *addr*	*addr*
001	Indirekt	LDA (*addr*)	$M(addr)$
010	Indexerad	LDA *disp*,(XR)	$XR + disp$
011	Relativ	JMP *disp*	$PC + 2 + disp$
100	Omedelbar	LDA #*n*	$PC + 1$
101	Underförstådd	INCA/INC	—

```
M(addr)->AR
M(M(addr))->AR
M(XR+disp)->AR
PC+2+disp->PC
n->AR
```

Exempelvis:



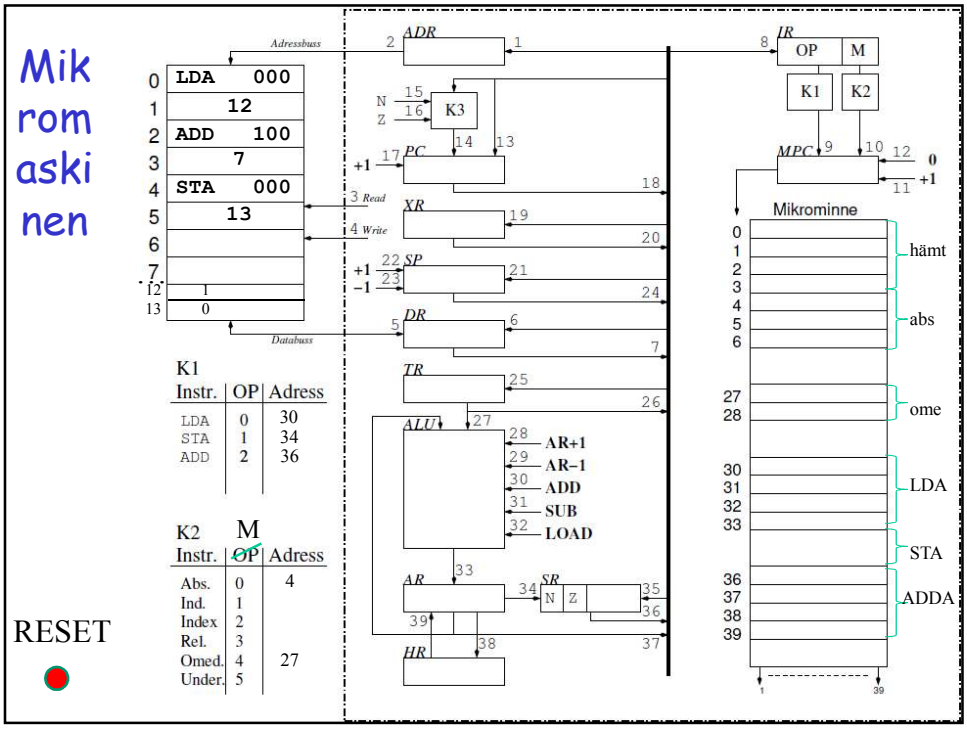
LDA 3

Absolut
EA=3
Operanden = 5

Instruktioner

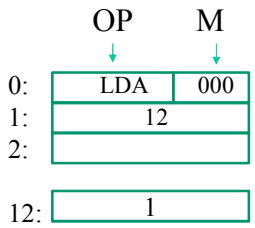
Instruktion	Verkan	Status N	Z	C	V
LDA *addr*	$AR := M(addr)$	*	*	-	0
STA *addr*	$M(addr) := AR$	-	-	-	0
ADD *addr*	$AR := AR + M(addr)$	*	*	*	*
SUB *addr*	$AR := AR - M(addr)$	*	*	*	*
INCA	$AR := AR + 1$	*	*	*	*
DEC	$AR := AR - 1$	*	*	*	*
CMP *addr*	$AR - M(addr)$	*	*	*	*
CLRA	$AR := 0$	0	1	0	0
ASRA	$AR := AR/2$	*	*	*	-
ASLA	$AR := AR \cdot 2$	*	*	*	*
LSRA	logiskt högerskift av AR	0	*	*	-
AND *addr*	$AR := AR$ and $M(addr)$	*	*	-	0
OR *addr*	$AR := AR$ or $M(addr)$	*	*	-	0
JMP *addr*	$PC := addr$	-	-	-	-
JMPN *addr*	$PC := addr$ om $N = 1$	-	-	-	-
JMPZ *addr*	$PC := addr$ om $Z = 1$	-	-	-	-
JMPC *addr*	$PC := addr$ om $C = 1$	-	-	-	-
JMPV *addr*	$PC := addr$ om $V = 1$	-	-	-	-
IN	$AR := IN$	*	*	-	0
OUT	$UT := AR$	-	-	-	0

Mik rom aski nen



RESET

Normal arbetsgång - översikt

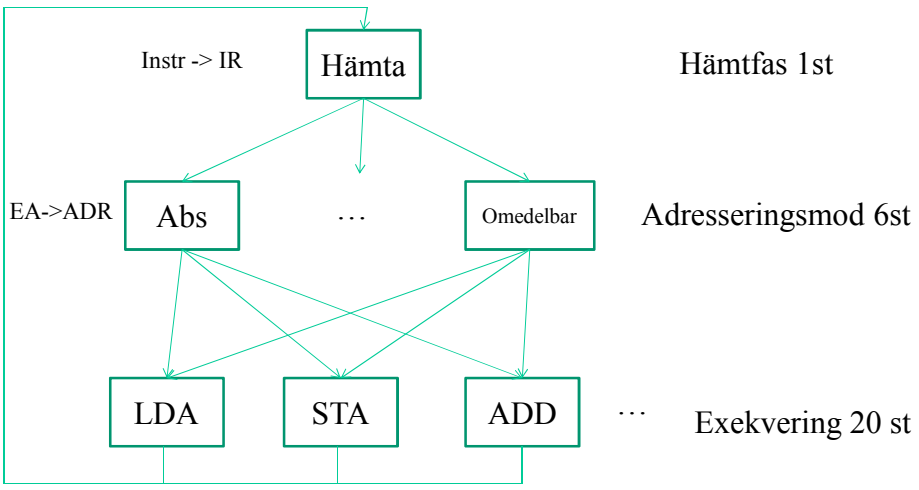


För varje instruktion {

1. **Hämtfas =>** Samma för alla instruktioner
   1. Hämta instruktionen till IR
   2. PC++
   3. Hoppa till rätt
2. **Adresseringsmodsfas** Beroende på M sker olika saker
   1. Vanligen: Hämta byten, PC++
   2. EA till ADR
   3. Hoppa till rätt
3. **Exekveringsfas =>** Beroende på OP sker olika saker
   1. Vanligen: Hämta operanden
   2. Resultatet till AR och uppdatera SR
   3. Hoppa till Hämtfas

}

Ritning: Organisation av mikroprogram



Steg 1 M(PC)->IR

Hämtfas

- 0: pc->adr, mpc++ 18,1,11
- 1: adr->minne,data->dr,mpc++ 2,3,5,11
- 2: dr->ir,mpc++ 7,8,11
- 3: PC++, K2->mpc 17,10

K2(0) = 4

Steg 2 M(PC)->ADR

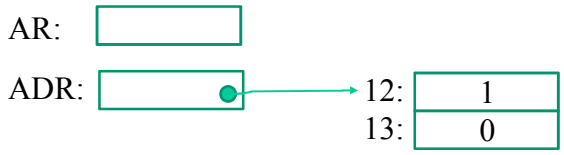
A-modsfas: absolut

- 4: pc->adr,mpc++ 18,1,11
- 5: data->dr,mpc++ 2,3,5,11
- 6: dr->adr,K1->mpc,PC++ 7,1,9,17

K1(0)=30

Steg 3 AR=M(ADR)

Exe-fas: LDA



- 30: data->dr,mpc++ 2,3,5,11
- 31: dr->tr,mpc++ 7,25,11
- 32: tr->ar,mpc++ 27,32,33,11
- 33: status, 0->mpc 34,12

Mikrokod för ADD #7

Steg 1 : samma som förut

K2(4) = 27

Steg 2: omedelbar

27: pc->adr, mpc++ 18,1,11
28: PC++,K1->mpc 17,9

K1(2)=36

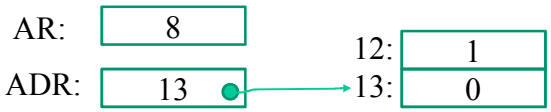
Steg 3: exekvering LDA AR = AR + M(ADR)

36: data->dr,mpc++ 2,3,5,11
37: dr->tr, mpc++ 7,25,11
38: ar+tr->ar,mpc++ 27,33,30,11
39: status, 0->mpc 34,12

Mikrokod för STA 13

Steg 1 : hämtfas har vi redan skrivit

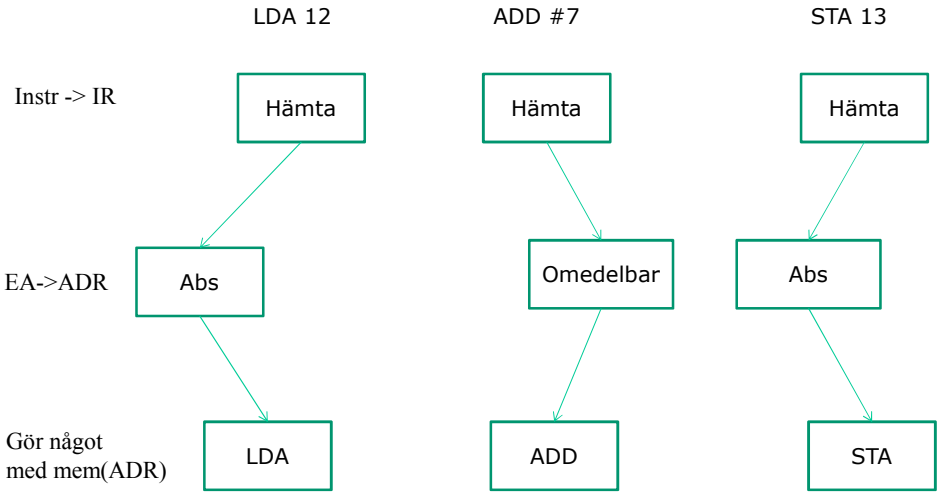
Steg 2: absolut har vi redan skrivit



K1(2)=34

Steg 3: exekvering	AR->M(ADR)
34: ar->dr,mpc++	37,6,11
35: dr->M, 0->mpc	2,4,5,12

Ritning

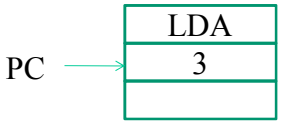


Mikrokod för LDA 3(X)

M(XR+3) ->AR

Hämtfas
finns redan

indexerad a-mod
12: PC->ADR, PC++, MPC++
13: M->DR, XR->TR, MPC++
14: DR->TR, TR->AR, AR->HR, MPC++
15: AR+TR->AR, MPC++
16: HR->AR, AR->**ADR**, K1->MPC



Exekvering för LDA
finns redan

Mikrokod för INCA

AR+1 ->AR

Hämtfas
finns redan

underförstådd a-mod
29: K1->MPC

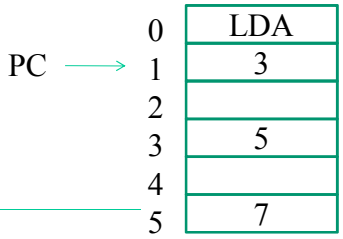
Exekvering för INCA
44: AR+1->AR, MPC++
45: status, 0->MPC

Mikrokod för LDA (3)

M(M(3)) ->AR

Hämtfas
finns redan

indirekt a-mod
7: PC->ADR, PC++, MPC++
8: M->DR, MPC++
9: DR->ADR, MPC++
10: M->DR, MPC++
11: DR->ADR, K1->MPC



Exekvering för LDA
finns redan