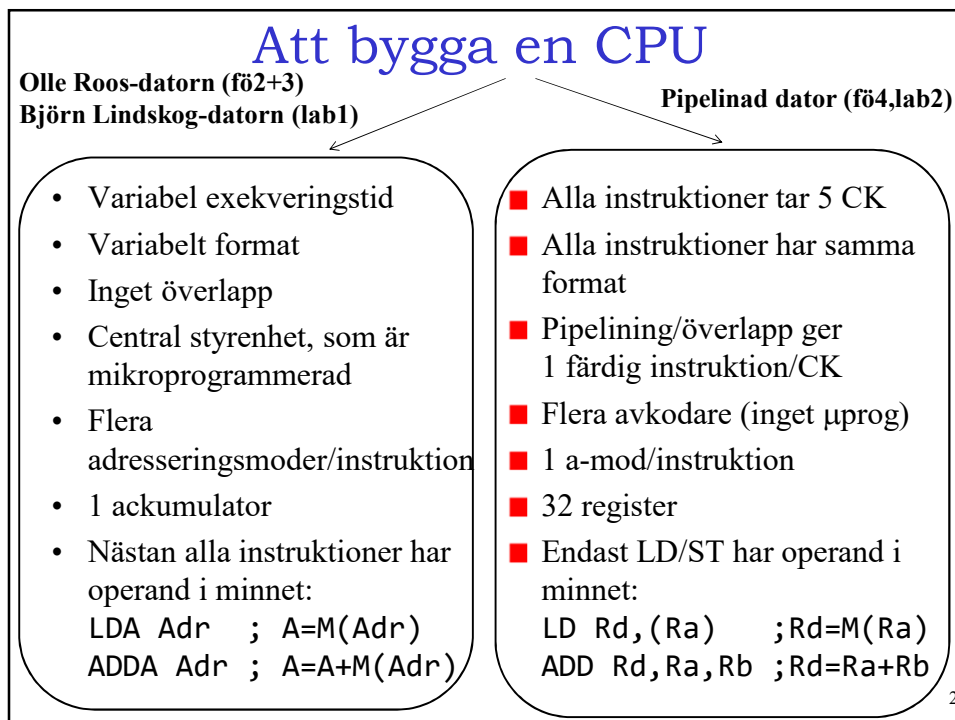


Mikroprogrammering I

1



2

Att bygga en CPU

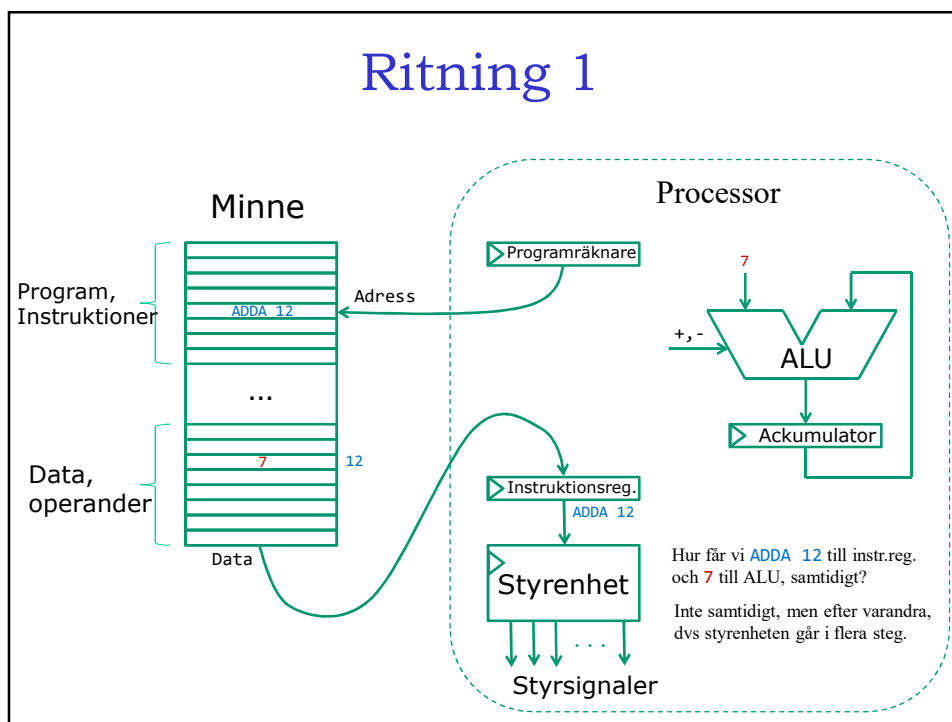
- Typisk CISC
- Programmering på 2 nivåer
asm och mikro
- Enkel controller:
 - garageportsöppnare
 - del av dator
- + Man kan göra avancerade instruktioner: **sortera**
- - Det blir många klockcykler / instr.

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

3

3

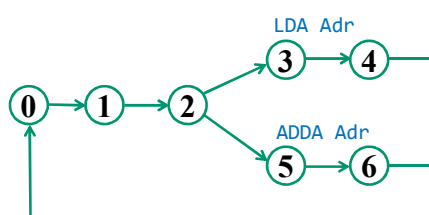
Ritning 1



4

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

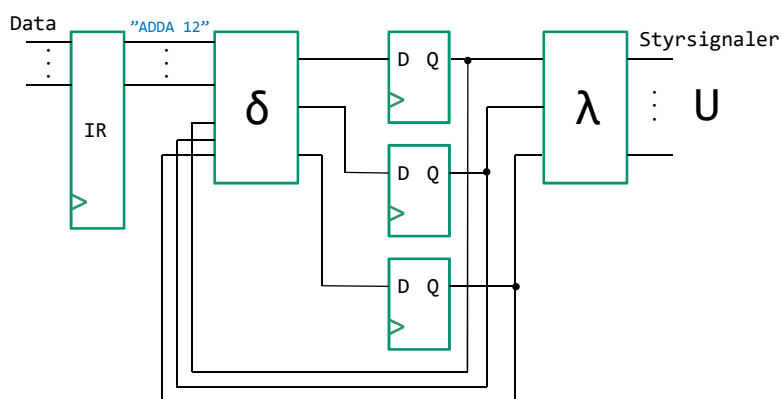


Tänkbara händelser vid olika steg:
 Steg 0-2: Hämta instruktion, samt beräkna adress
 Steg 3-4: Utför en instruktion
 Steg 5-6: Utför en annan instruktion

5

Mikroprogrammering

Vi skulle kunna bygga sekvensnätet på "vanligt" sätt:

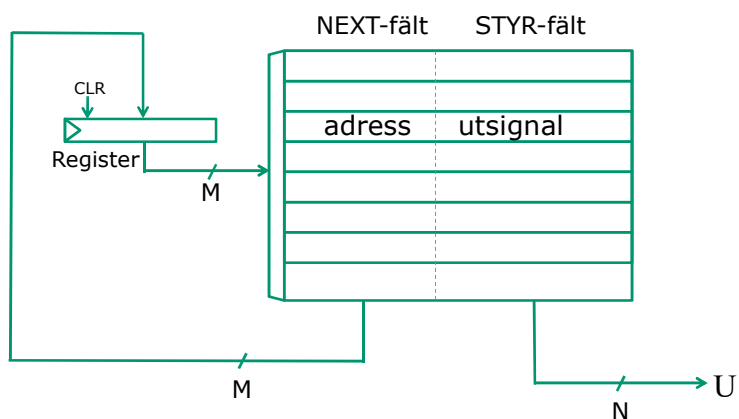


Men det blir väldigt jobbigt för 100+ tillstånd, dessutom omständligt att konfigurera om vid behov.

6

Ritning2

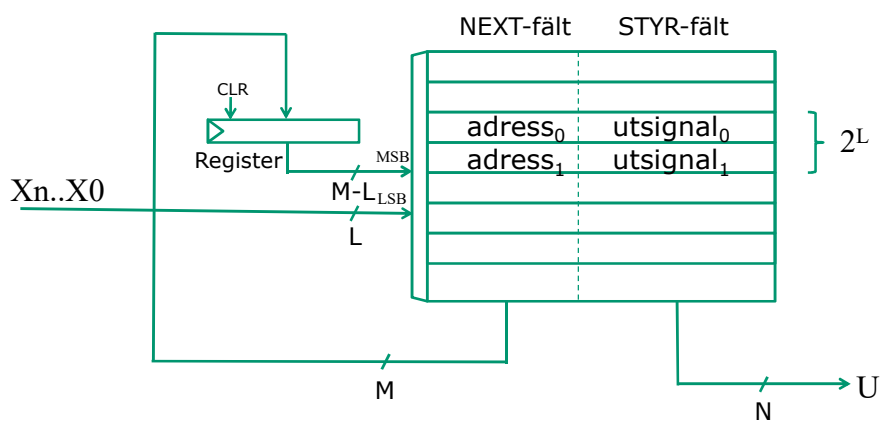
Autonom styrenhet med ROM/Register



7

Variant2

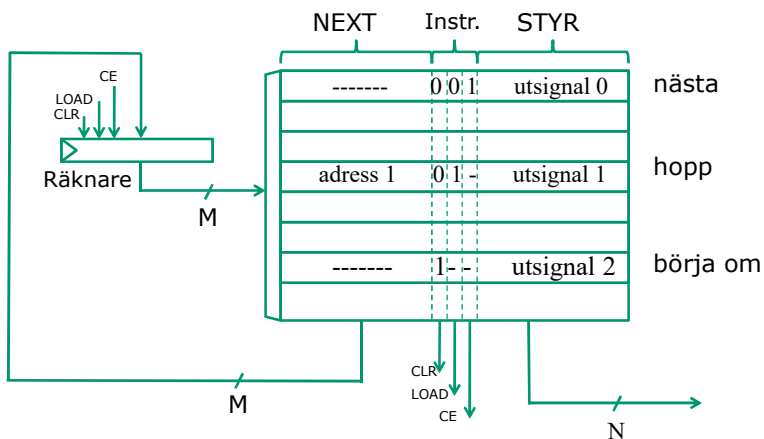
Styrenhet med ROM/Register



8

Ritning 3

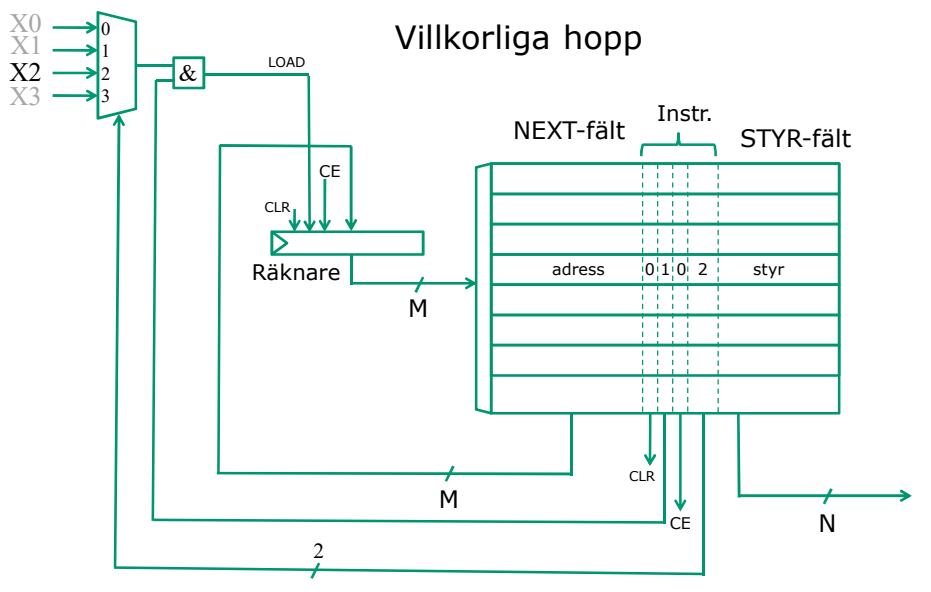
Autonom styrenhet med ROM/Räknare



9

Ritning 4

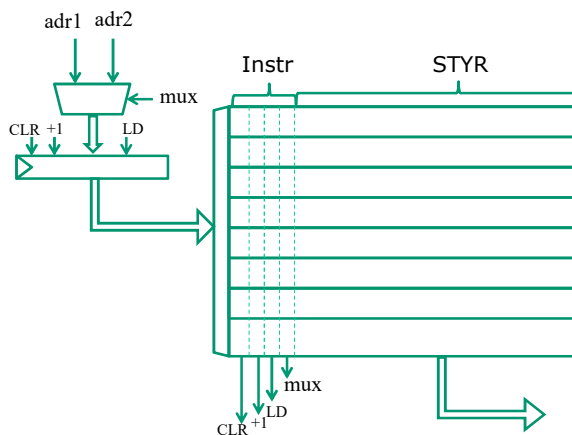
Villkorliga hopp



10

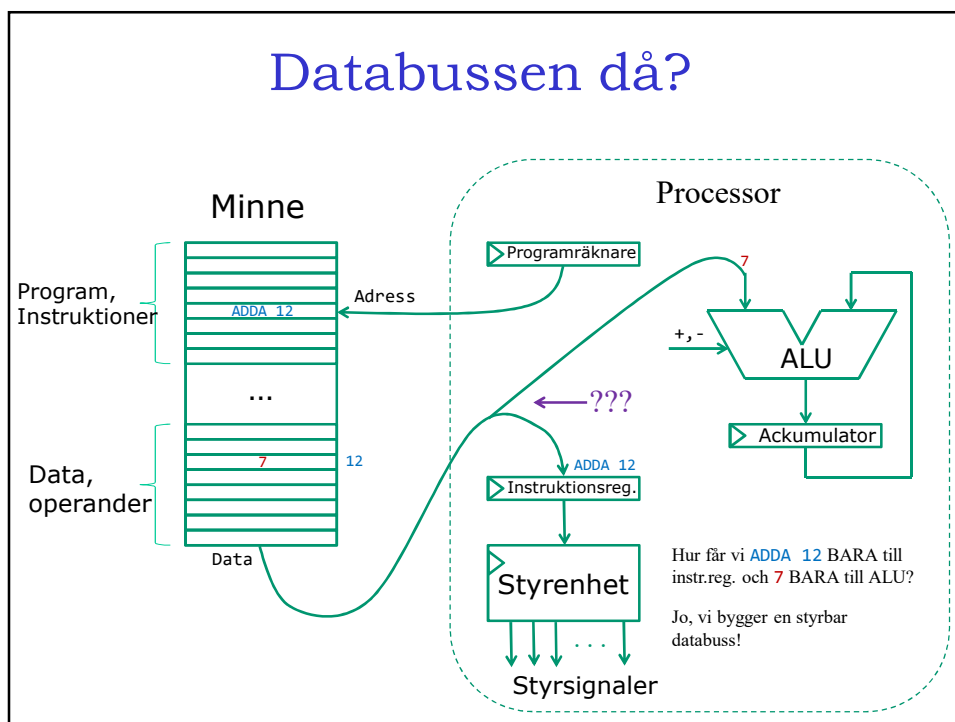
En variant

Autonom styrenhet utan adressfält

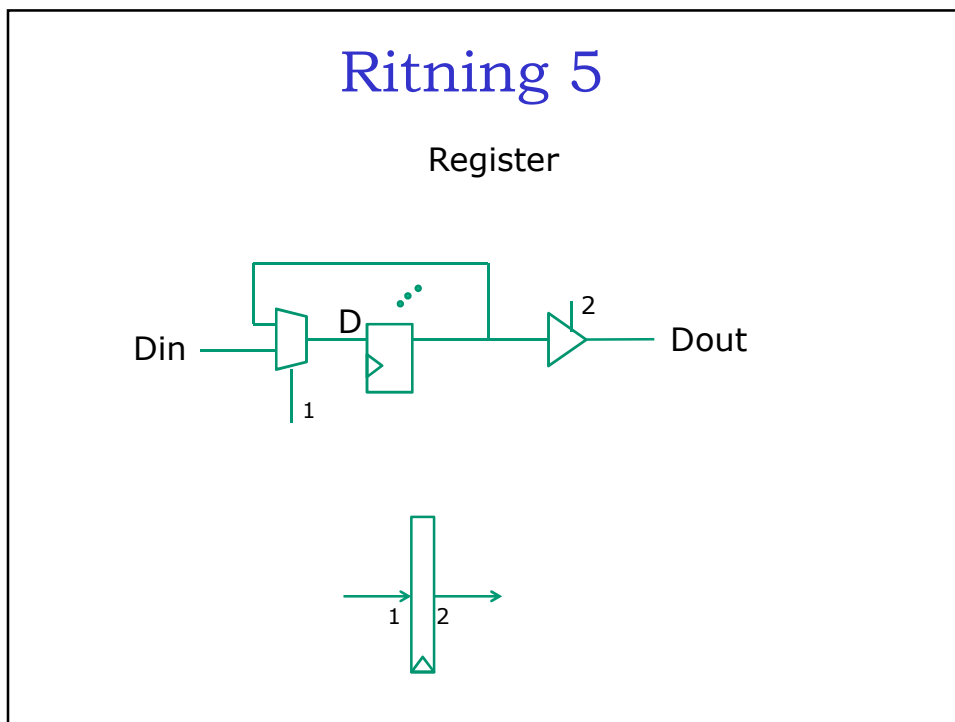


11

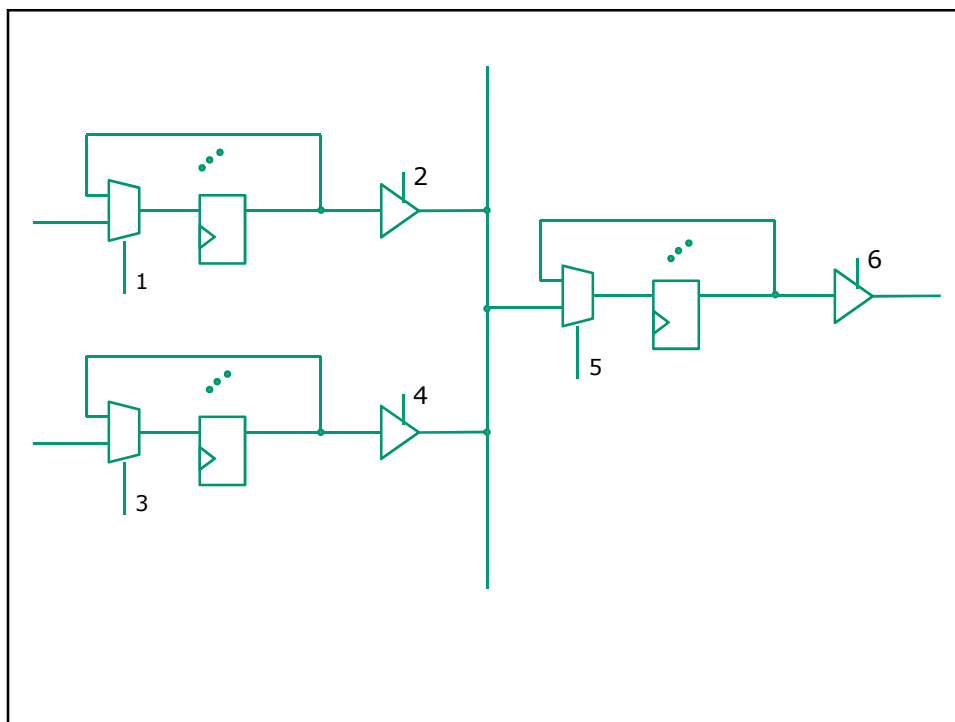
Databussen då?



12

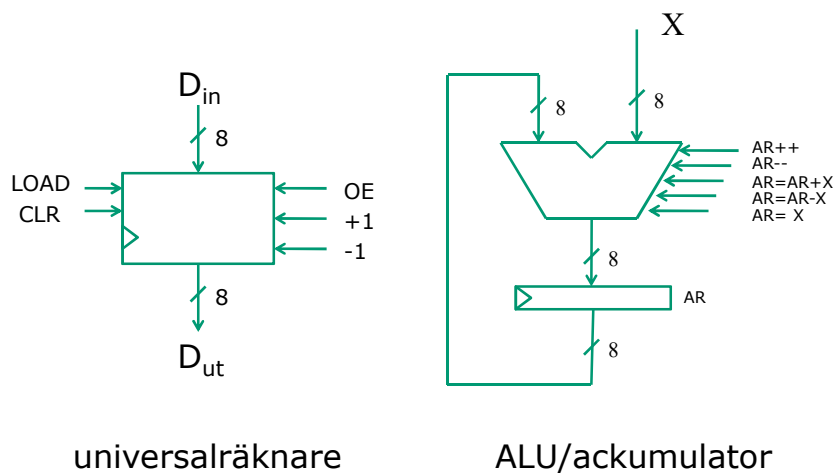


13



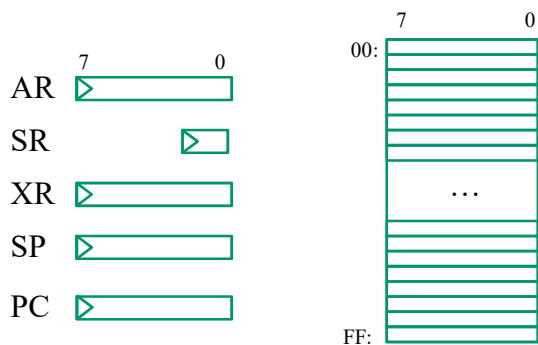
14

Ritning 6 och 7



15

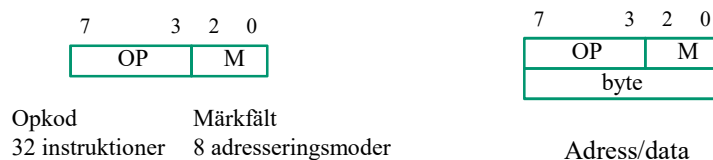
Ritning 8 - Programmerarmodell



Endast 2 flaggor: Z,N

16

Ritning 9 - instruktionsformat



17

Adresseringsmoder

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

Exempelvis:

LDA 3

| | | |
|----|-----|-----|
| 0: | LDA | 000 |
| 1: | | 3 |
| 2: | | |
| 3: | | 5 |

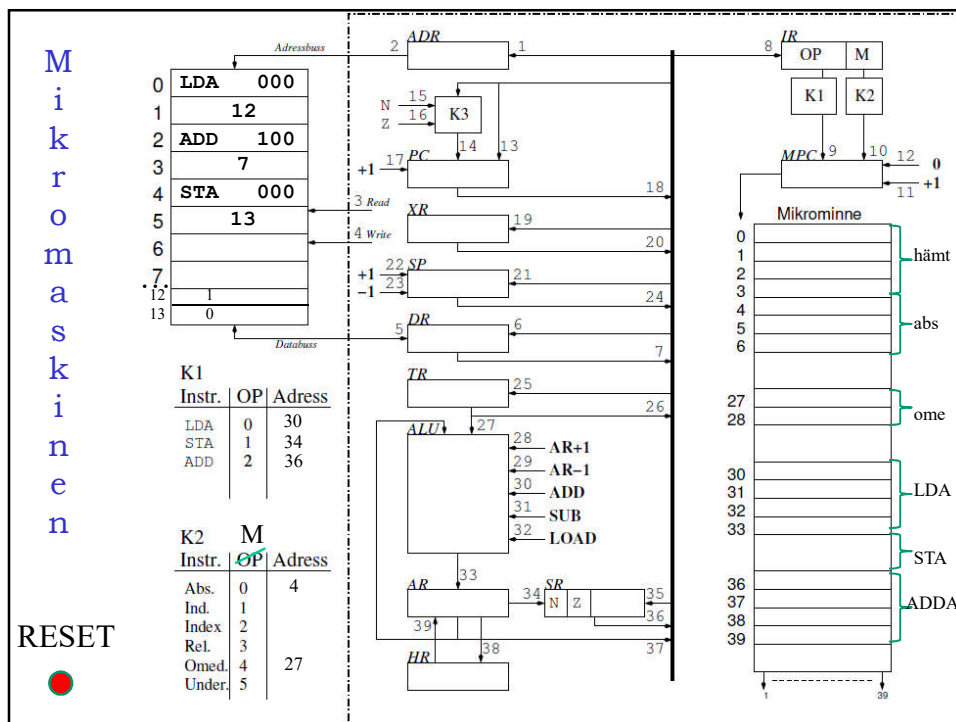
Absolut
EA=3 (Effektiv Adress)
Operanden = 5

18

Instruktioner

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

19



20

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

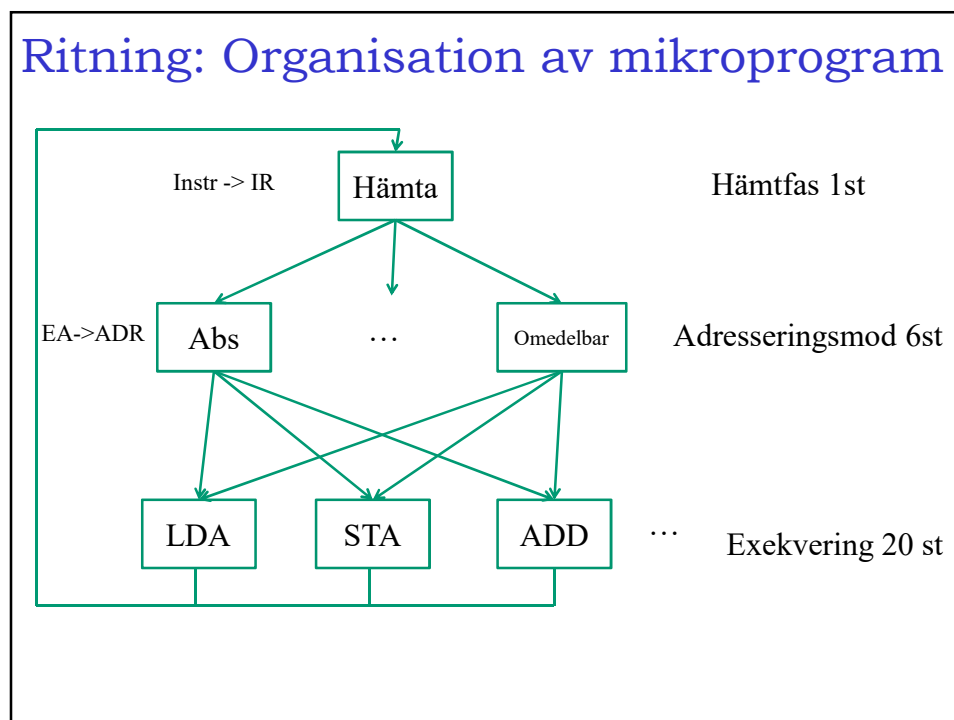
1. Vanligen: Hämta operanden
2. Resultatet till AR och uppdatera SR
3. Hoppa till Hämtfas

}

| | OP | M |
|-----|-----|-----|
| 0: | LDA | 000 |
| 1: | 12 | |
| 2: | | |
| 12: | 1 | |

21

Ritning: Organisation av mikroprogram



22

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

23

Steg 2 M(PC)->ADR

A-modsfas: absolut



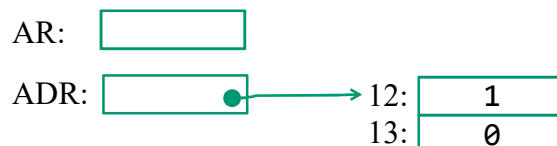
- 4: pc->adr,mpc++ 18,1,11
- 5: data->dr,mpc++ 2,3,5,11
- 6: dr->adr,K1->mpc,PC++ 7,1,9,17

K1(0)=30

24

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

25

Mikrokod för ADD #7

Steg 1 : samma som förut

$K2(4) = 27$



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

$K1(2)=36$

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

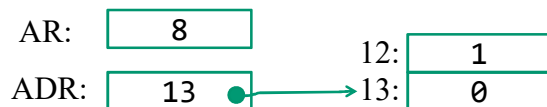
- 36: data->dr, mpc++ 2, 3, 5, 11
37: dr->tr, mpc++ 7, 25, 11
38: ar+tr->ar, mpc++ 27, 33, 30, 11
39: status, 0->mpc 34, 12

26

Mikrokod för STA 13

Steg 1 : hämtfas har vi redan skrivit

Steg 2: absolut har vi redan skrivit



$K1(2)=34$

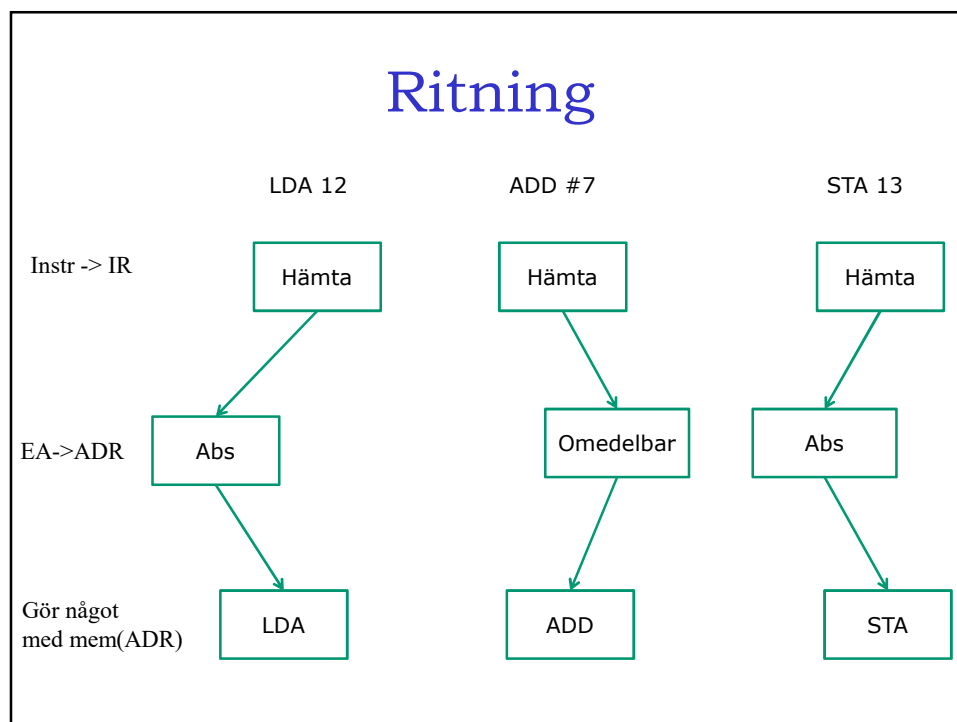
Steg 3: exekvering AR->M(ADR)

34: ar->dr, mpc++ 37, 6, 11

35: dr->M, 0->mpc 2, 4, 5, 12

27

Ritning



28

Mikrokod för LDA 3(X)

$M(XR+3) \rightarrow AR$

Hämtfas

finns redan

indexerad a-mod

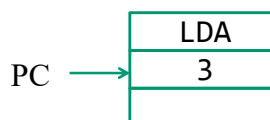
12: PC \rightarrow ADR, PC++, MPC++

13: M \rightarrow DR, XR \rightarrow TR, MPC++

14: DR \rightarrow TR, TR \rightarrow AR, AR \rightarrow HR, MPC++

15: AR+TR \rightarrow AR, MPC++

16: HR \rightarrow AR, AR \rightarrow ADR, K1 \rightarrow MPC



Exekvering för LDA

finns redan

29

Mikrokod för INCA

$AR+1 \rightarrow AR$

Hämtfas

finns redan

underförstådd a-mod

29: K1 \rightarrow MPC

Exekvering för INCA

44: AR+1 \rightarrow AR, MPC++

45: status, 0 \rightarrow MPC

30

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

| | | |
|------|---|-----|
| | 0 | LDA |
| PC → | 1 | 3 |
| | 2 | |
| | 3 | 5 |
| | 4 | |
| | 5 | 7 |

Exekvering för LDA

finns redan