

Datorteknik TSEA82 + TSEA57

Fö1

Introduktion till kursen

Datorteknik Fö1 : Agenda

- Informationskanaler
- Introduktion till kursen Datorteknik
- Fö1 : Datormodellen, Programmerarmodellen, Instruktioner
- Nödvändiga programvaror
- Tid för frågor

Informationskanaler

Informationskanaler

- Lisam : Kursens sidor ligger i Lisam (använd inte ISY-websidorna)
- Microsoft Teams : Används till hjälp med laborationer utanför labbtid
- Mail + Teams : Används för övrig kommunikation

Introduktion till kursen Datorteknik

Vad är datorteknik, i den här kursen?

Assemblerinstruktioner

Subrutiner

Binär aritmetik

Assemblerprogrammering

Adresseringsmoder

Digitalteknik

ATmega16

Stacken

A/D- D/A-omvandlare

Avbrott

Samläsning D & I

D : TSEA82, 4hp 4 labbar (+labb 0)

I : TSEA57, 6hp 5 labbar (+labb 0)

Labbar + LAX = Godkänd kurs

Godkända

Labb : ca 95%

LAX är Lab-eXamination!

Lax : ca 80%

Vad händer när?

V13 V14 V15 V16 V17 V18 V19 V20 V21

Fö1	Fö4	Fö6	Fö7	Fö8	Fö9	Le		TPVT2
Fö2	La0	La1	La2		La3	La4	La5	LAX!
Fö3	Fö5							

Vad händer sen?

TSEA57 → TSEA56 : Elektronik kandidatprojekt (vt2024)

TSEA82 → TSEA83 : Datorkonstruktion (vt2024)

→ TSEA29 : Konstruktion med mikrodator (ht2024)

Resultat från föregående års utvärdering

Resultat 2022

Hur många läste kursen TSEA82/TSEA57 : 108 / 11

Hur många svarade på utvärderingen : 19 / 3

Helhetsbetyg : 4,53 / 5,00

Förändringar inför årets kurs:

-Inga förändringar

Lisamsida + kursmateriel

...

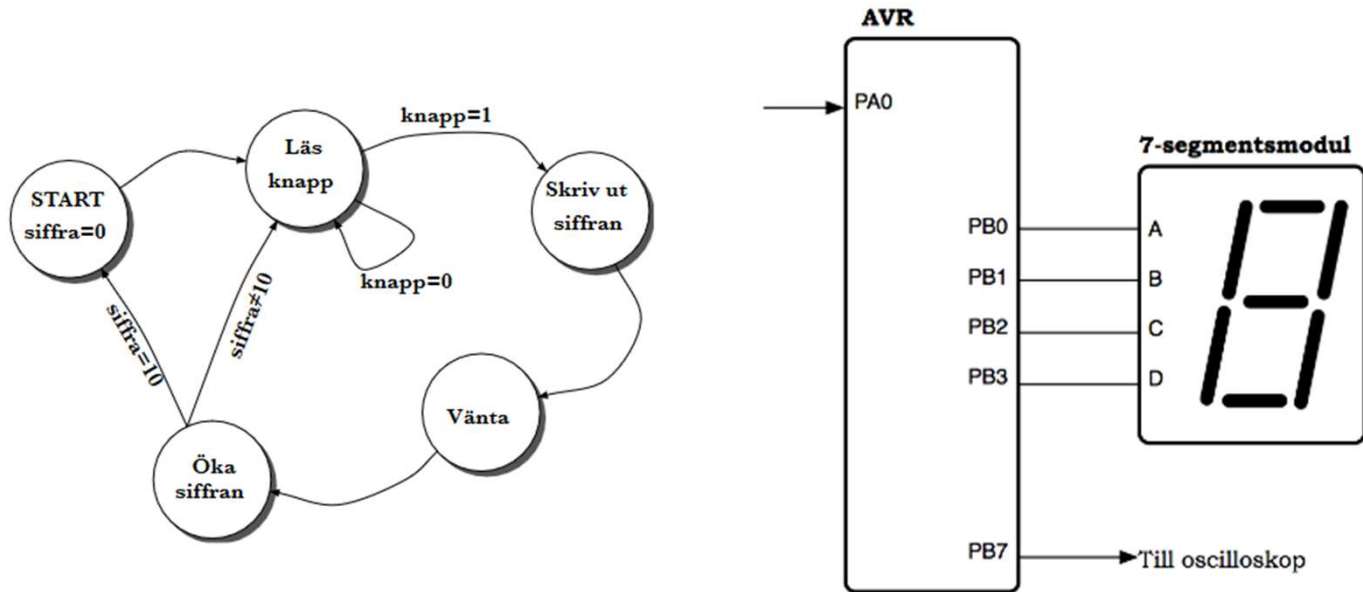
Laborationer

Laborationer

- Alla laborationer görs parvis
- Labb0 är till för att bekanta sig med labbmiljön
- Labb1->Labb4 är ordinarie labbar
- Labb5 är endast för I-programmet
- Anmälan till alla tillfällen görs i Lisam

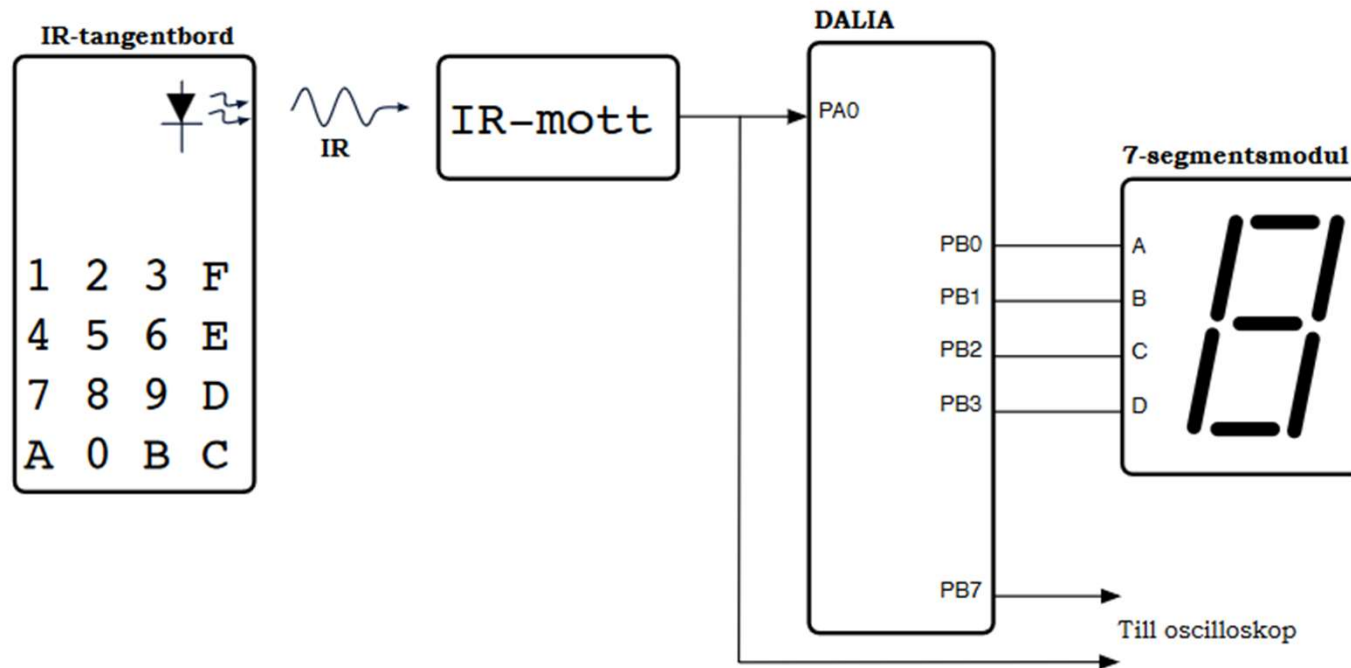
Labb0

- Atmel Studio + Miniprojekt



Labb1

- IR-länk



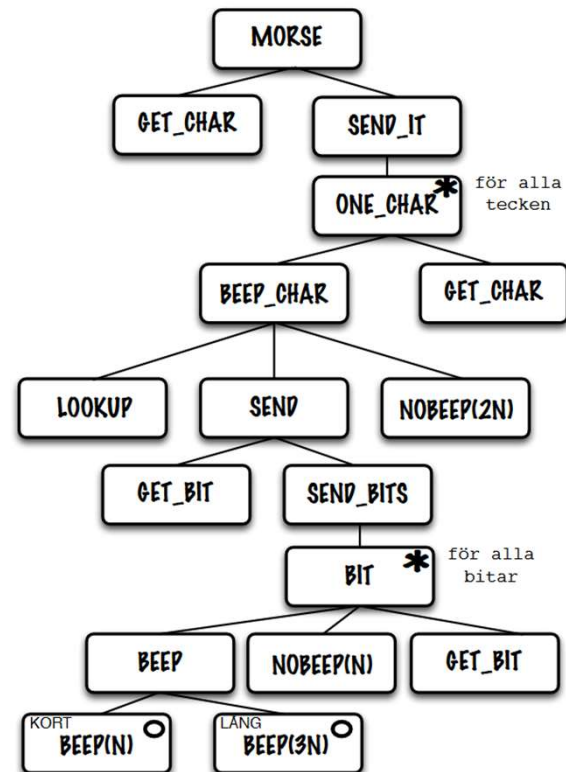
Moment:

- Enklare assembler-programmering
- Timing

Lab2

- Signalering av morse-kod

A	• —	V	• • • —
B	— • • •	W	— • —
C	— • • — •	X	— • • —
D	— • •	Y	— • — —
E	•	Z	— — • •
F	• • — •	.	• • — • • •
G	— — • •	,	— — — — —
H	• • • •	?	• • — • • •
I	• •	/	— — — • •
J	— — — — —	@	• — — • • •
K	• • — •	1	• — — — —
L	• • • •	2	• • — — —
M	— — —	3	• • • — —
N	— • •	4	• • • —
O	— — — —	5	• • • •
P	• — — • •	6	— • • • •
Q	— — • • —	7	— • • • •
R	• • — •	8	— — — • •
S	• • •	9	— — — — •
T	— •	0	— — — — —
U	• • —		

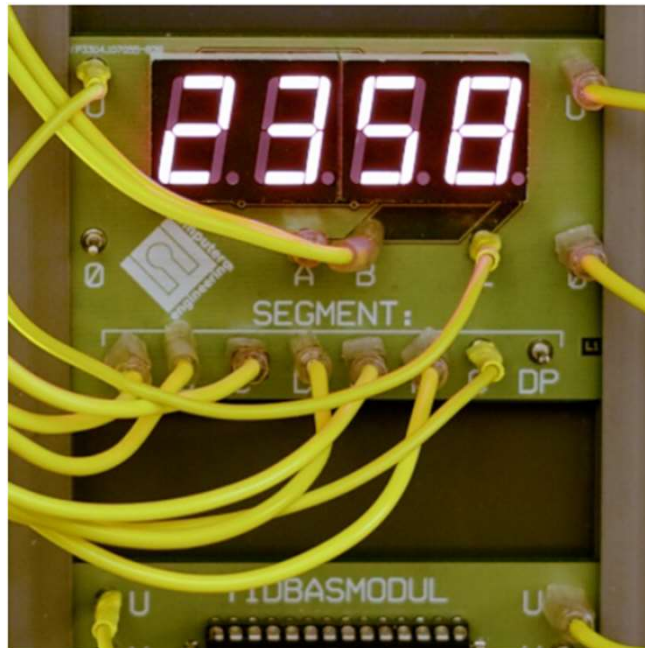


Moment:

- Subrutiner
- Timing

Labb3

- Digital-ur

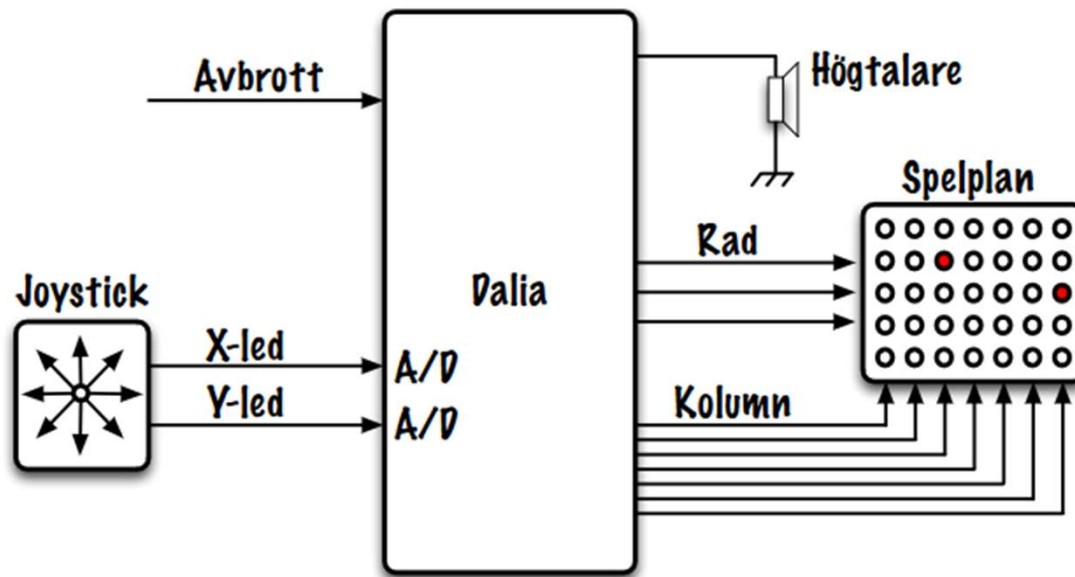


Moment:

- Avbrott
- Multiplexning
- Subrutiner
- Timing

Lab4

- Spel

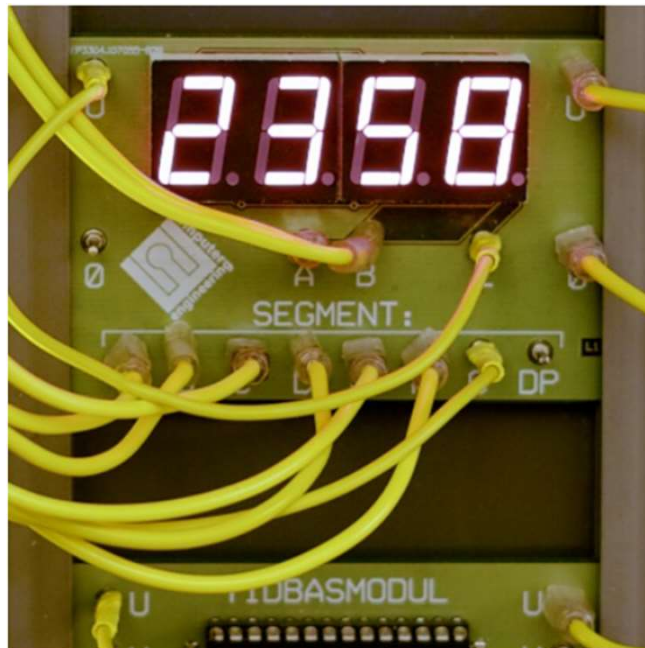


Moment:

- A/D-omvandling
- Avbrott
- Multiplexning
- Subrutiner
- Timing

Labb5, endast I-programmet

- C-programmering (digitalur + skenbar parallellism)



Moment:

- C-programmering
- Timers
- Parallellism
- Avbrott
- Multiplexning

LAX

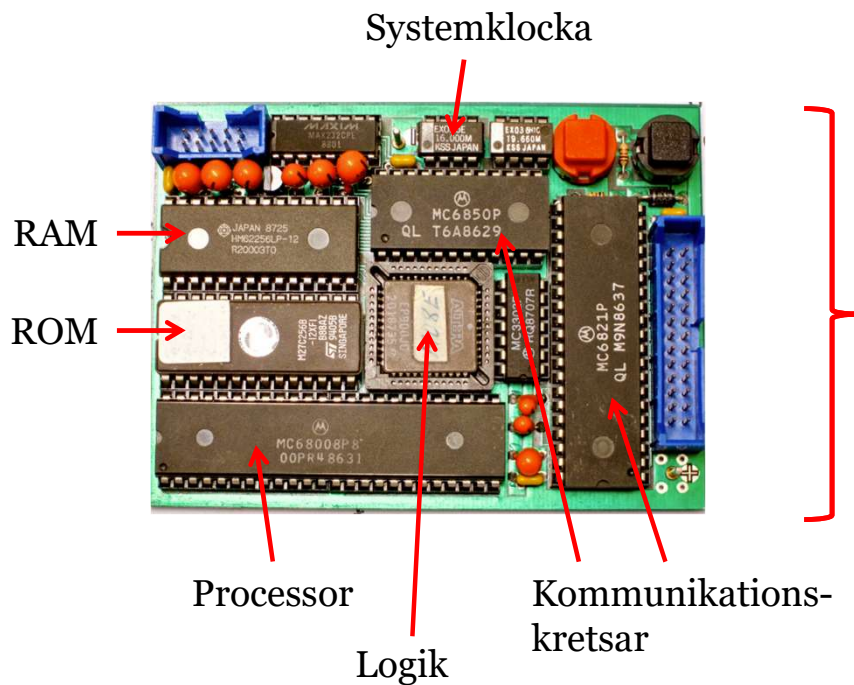
LAX (Lab-eXamination)

- LAX:en görs individuellt, utan hjälp
- Uppgiften ska redovisas inom 90 minuter
- En enklare mindre uppgift
- Observera! Även om LAX:en går vid flera tillfällen så kan man bara gå på ett av dessa tillfällen

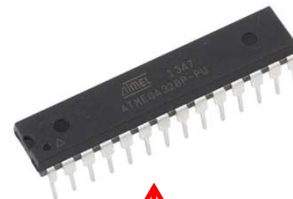
Datormodellen + Programmerarmodell

Utveckling

Tutor (1985-2015)

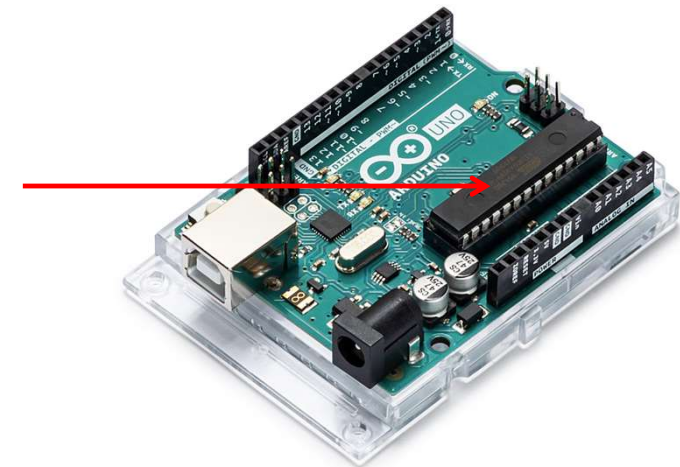


ATmega16
mikrokontroller



Allt på ett chip!
(och lite till)

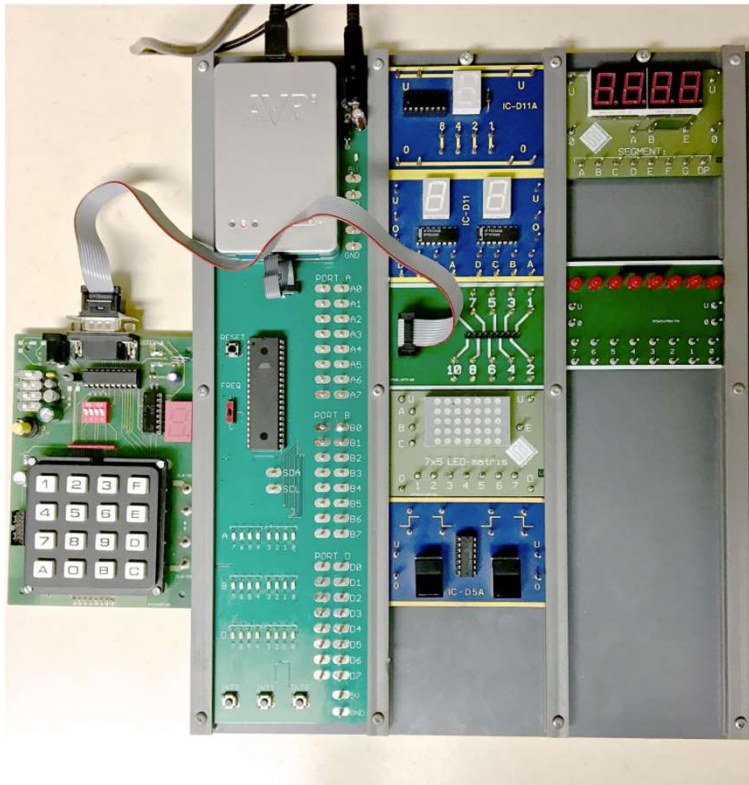
Arduino UNO (ATmega328)



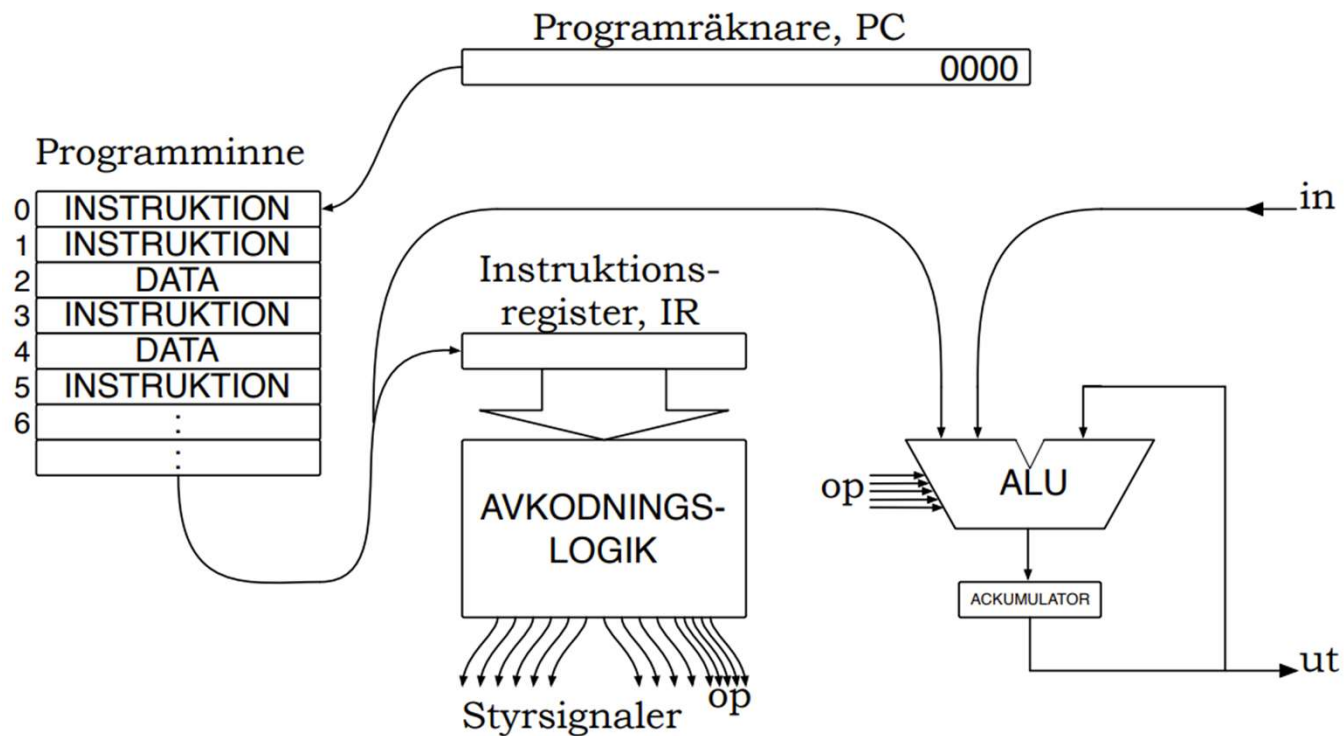
Ett av *många* användningsområden!

Dalia

Vi använder Dalia och processorn ATmega16.



Datormodell (grundläggande)



Nödvändiga delar

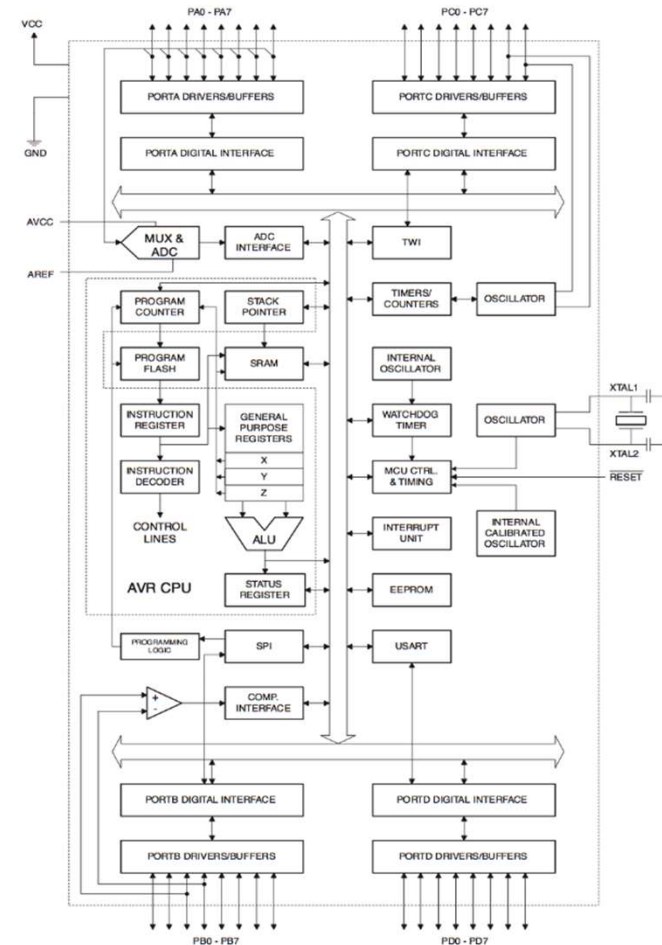
- Programräknare
- Programminne
- Instruktionsregister
- Avkodning+styrsignaler
- ALU
- In/Ut-enheter

Datormodell (ATmega16)

Utöver CPU:ns nödvändiga delar

- EEPROM
- Flera timers
- USART
- SPI
- TWI
- Komparator
- Debug-logik
- Watchdog
- m m ...

Används inte
i den här kursen,
MEN kommer till
användning i senare
kurser.



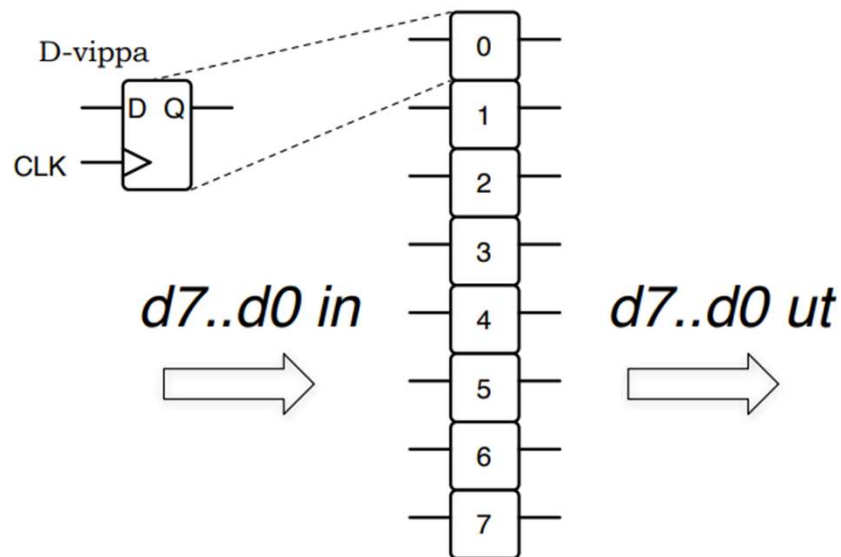
Programmerarmodell

Processorns grundläggande delar, registeruppsättning, programräknare och statusregister brukar kallas processorns *programmerarmodell*.

För ATmega16 blir det:

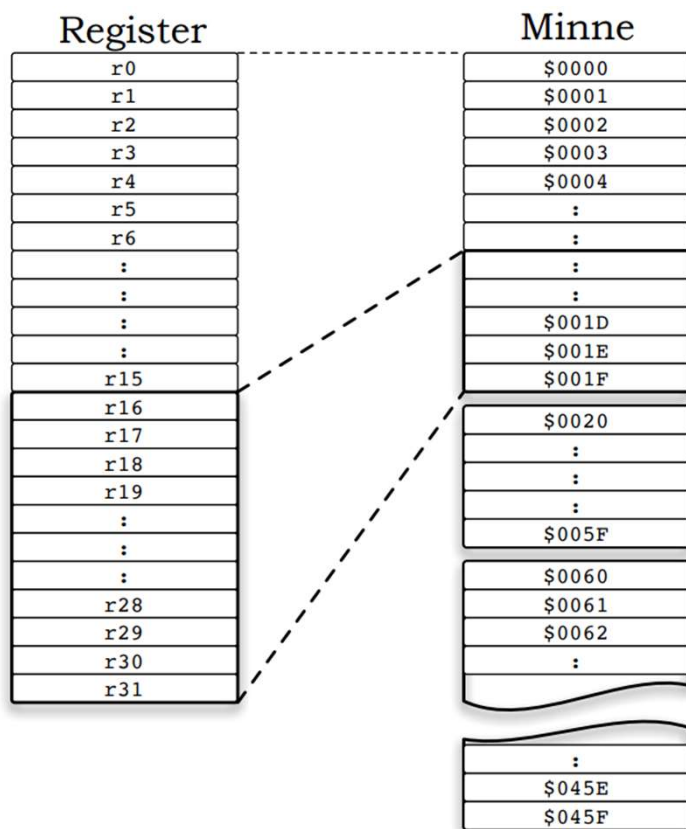
- Arbetsminnet, SRAM (för temporär lagring)
- Generella register, r0-r31 (för generell användning)
- IO-register (register med specifik användning)
- Programräknare (adress till nuvarande instruktion)
- Statusregister (information om det senaste resultatet från ALU:n)

Programmerarmodell : Generella register (r0-r31)



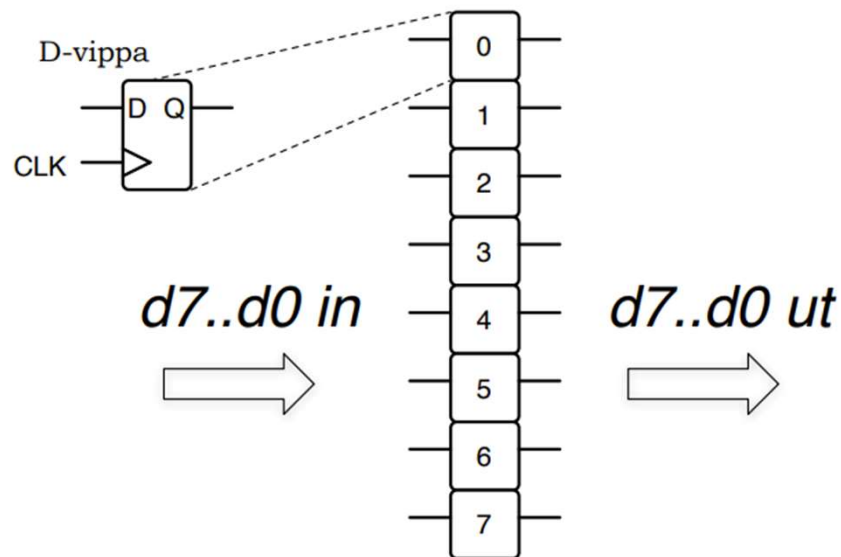
- Samtliga generella register är 8 bitar breda
- Generella register r0-r31 kan användas för godtyckligt syfte. MEN "bara" r16-r31 kan användas för alla processorns aritmetiska och logiska operationer.¹ Bara vissa operationer fungerar för r0-r15.
- Vissa generella register, r24-r31, kan kombineras parvis två och två, för att bilda 16 bitar breda register.
[r25:r24], [r27:r26], [r29:r28], [r31:r30]

Programmerarmodell : Generella register (minnesmappning)



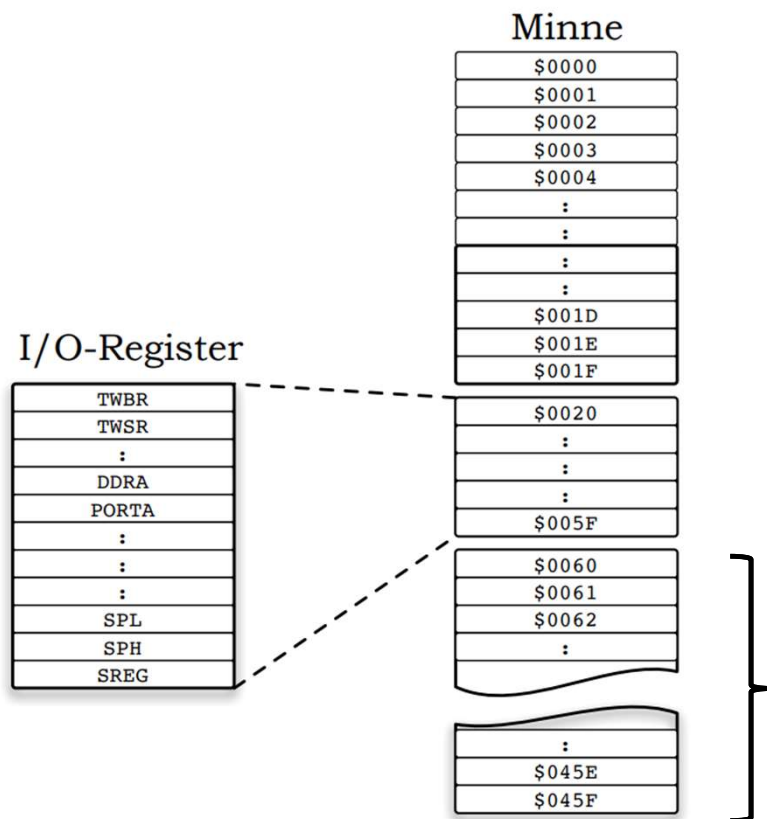
- Samtliga register har en motsvarande minnesadress
- Skriver man något till t ex register r7, så finns det tillgängligt att läsa på adress \$0007.
- P s s, skriver man något till t ex adress \$0002, så finns det tillgängligt att läsa i register r2.

Programmerarmodell : I/O-register



- Samtliga I/O-register är 8 bitar breda
- I/O-register har ett specifikt syfte. T ex att styra datariktningen på en viss port (DDRB), eller att sätta portens utvärde (PORTB), eller för att läsa portens invärde (PINB).
- Alla register som inte är generella register, räknas som I/O-register, även om de inte har med in- eller ut-data att göra.
- Vissa I/O-register, t ex stackpekarens SPH och SPL, kombineras parvis två och två, för att bilda 16 bitar breda register. $SP=SPH:SPL$

Programmerarmodell : I/O-register (minnesmappning)



- Samtliga I/O-register har en motsvarande minnesadress
- Skriver man något till t ex I/O-register PORTB, så finns det tillgängligt på adress \$0025.
- P s s, skriver man något till just adress \$0025, så finns det tillgängligt på PORTB.

Programmerarmodell : Arbetsminne (SRAM)

Minne

\$0000
\$0001
\$0002
\$0003
\$0004
:
:
:
:
\$001D
\$001E
\$001F
\$0020
:
:
:
\$005F
\$0060
\$0061
\$0062
:
:
\$045E
\$045F

- Adresserna \$0000-\$05F är bara en mappning (en koppling) till något register, generellt register eller I/O-register.
- Det egentliga arbetsminnet (SRAM) finns mellan adresserna \$0060-\$045F (gäller ATmega16)

Programmerarmodell : Statusregistret (SREG)

SREG

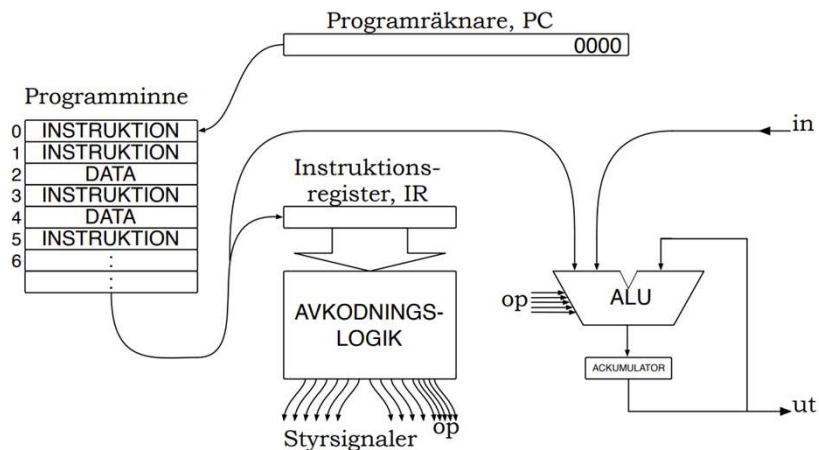
I	T	H	S	V	N	Z	C
---	---	---	---	---	---	---	---

Flagga	Innebörd
--------	----------

Z	1 om resultatet lika med noll
N	1 om resultatet mindre än noll, det vill säga mest signifikant bit satt
C	1 om <i>Carry</i> , det vill säga minnesbiten vid beräkningar vid teckenlösa tal satt
V	1 om <i>overflow</i> , om felaktigt resultat vid beräkningar av 2-komplementkodade tal

- Statusregistret innehåller information, så kallade flaggor, om det senaste resultatet från t ex en aritmetisk eller logisk operation. Dvs, resultatet från den aritmetiska logiska enheten, ALU:n.
- Flaggorna Z, N, C och V är mest intressanta.

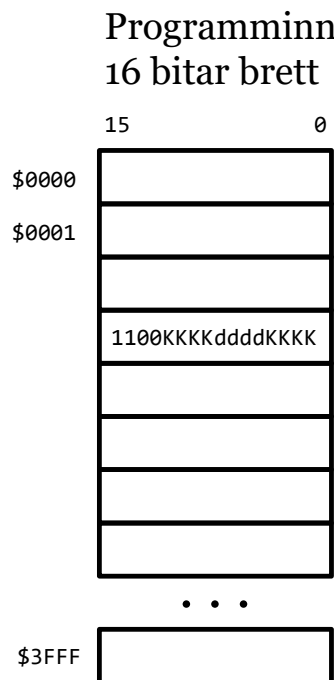
Programmerarmodell : Programräknaren (PC)



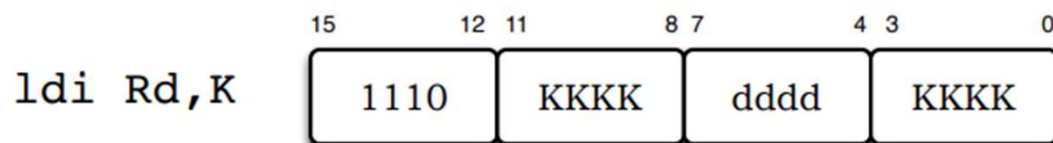
- Programräknaren innehåller adressen till den instruktion som utförs för tillfället.
- När programmet kör rad för rad kommer PC att räknas upp allteftersom.
- Om programmet gör ett hopp kommer PC att laddas med en ny adress, dvs dit man hoppar.

Instruktioner (början)

Instruktionsformat



- Instruktioner lagras i programminnet i ett instruktionsformat.
- Det finns flera instruktionsformat, dvs det varierar beroende på instruktion.
- Programminnet är 16 bitar brett
- Instruktionen `ldi r16,18` skulle t ex använda sig av nedanstående instruktionsformat och lagras som 1110 0001 0000 0010:



Instruktioner

I databladet för mikrokontrollern finns drygt hundralet instruktioner. Dessa kan delas in i fem huvudgrupper:

- Grupp 1. Instruktioner som flyttar data (`ldi`, `mov`, ...)
- Grupp 2. Aritmetiska instruktioner (`add`, `sub`, `mul`, ...)
- Grupp 3. Logiska instruktioner (`asl`, `ror`, ...)
- Grupp 4. Hoppinstruktioner (`jmp`, `brxx`, `call`, ...)
- Grupp 5. I/O-instruktioner (`out`, `in`)

Instruktioner : Grupp 1. Flytta data¹

Mellan generella register används instruktionen mov.

Exempel: mov

Flytta (kopiera) innehållet i register r16 till r13

mov r13,r16 ; Load immediate

↑ ↑ ↑ ↑

instruktion destinationsregister källregister kommentar

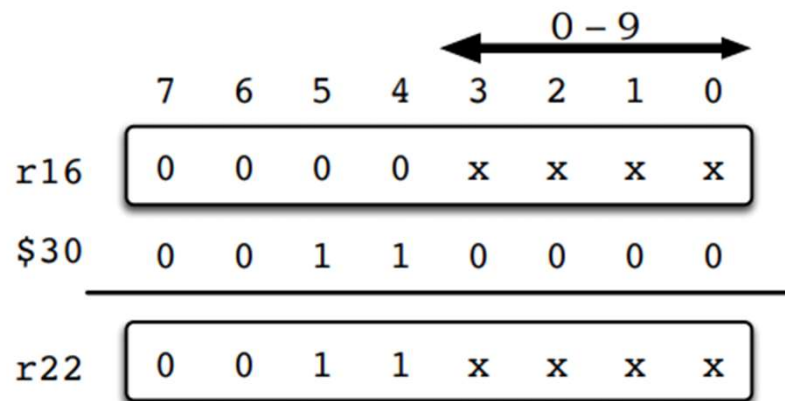
Instruktioner : Grupp 1. Flytta data

8-bitars register, kan inneha tal mellan 0-255 (00000000-11111111)

Exempel: Decimalsiffror till ASCII¹

Register r16 innehåller en siffra, 0-9.

Översätt den till ASCII¹



```
ldi    r22,$30
```

```
add     r22,r16
```

alternativt

```
mov     r22,r16
```

```
ori     r22,$30
```

Instruktioner : Grupp 1. Flytta data

16-bitars register, kan inneha tal mellan 0-65535

Exempel: Öka på r24:r25 med ett steg

adiw fungerar enbart för registerparen r25:r24, r27:r26, r29:r28, r31:r30.

```
adiw r24,1 ; r25 underförstås
```

Exempel: Öka på r17:r16 med ett steg

```
inc r16 ; affects Z but not C
```

```
brne DONE
```

```
inc r17
```

```
DONE:
```

Instruktioner : Grupp 1. Flytta data

16-bitars register, kan inneha tal mellan 0-65535

För att peka ut en adress i arbetsminnet (\$0060-\$045F) krävs mer än 8 bitar, då kan man använda ett 16-bitars pekarregister, X, Y eller Z, som motsvaras av registerparen r27:r26, r29:r28 respektive r31:r30.

Exempel: Invertera talet som finns på adress \$0102

Med pekare:

```
ldi    ZH,HIGH($0102) ; ZH(r31)=01
ldi    ZL,LOW($0102)  ; ZL(r30)=02
ld     r16,Z           ; r16=Mem(Z)
com     r16             ; complement r16
st     Z,r16           ; Mem(Z)=r16
```

Utan pekare:

```
lds     r16,$0102      ; r16=Mem($0102)
com     r16             ; complement r16
sts     $0102,r16      ; Mem($0102)=r16
```

Nödvändiga programvaror

Nödvändiga programvaror

För den som vill jobba på egen dator:

- Atmel Studio, kursen använder version 7
- Om man inte har Windows:
 - Virtual Box (Mac/Linux)
 - Instruktioner finns i Lisam, under Distansarbete

Vid laborationerna vill vi dock att man redovisar på våra datorer i labbet.

Tid för Frågor

Anders Nilsson

www.liu.se