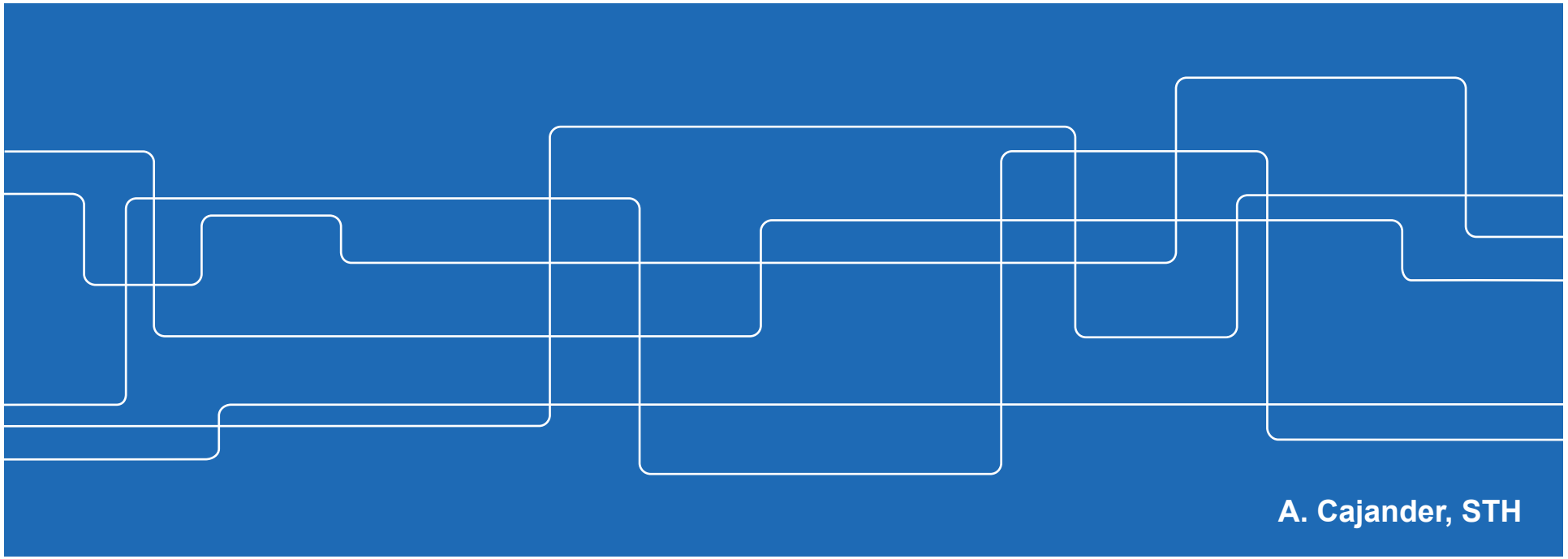
HF0010

Introduktionskurs i Datateknik

F2: Vad är ett programmeringsspråk?



A. Cajander, STH

Introduktionskurs i Datateknik

F1: Vad är en dator?

F2: programmeringsspråk?

L1: Hour of code, alla kan programmera, Diplom!

L2: Visuell programmering av svårare problem!

F3: Webbprogrammering – JavaScript!

L3: IDE, Web sida & JavaScript!

HTML

Kommer du ihåg?!

SW

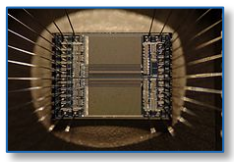
PROGRAM
KOMPILATOR
OPERATIVSYSTEM

Källkod ASSEMBLER Assemblerkod

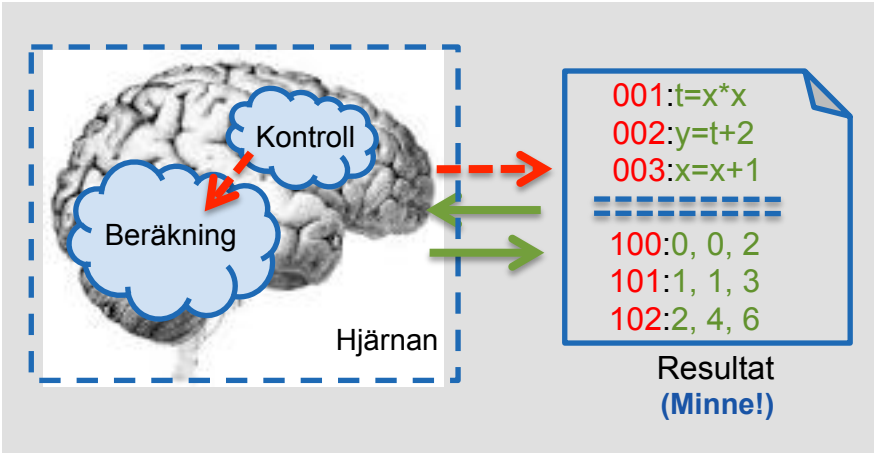
Maskinkod

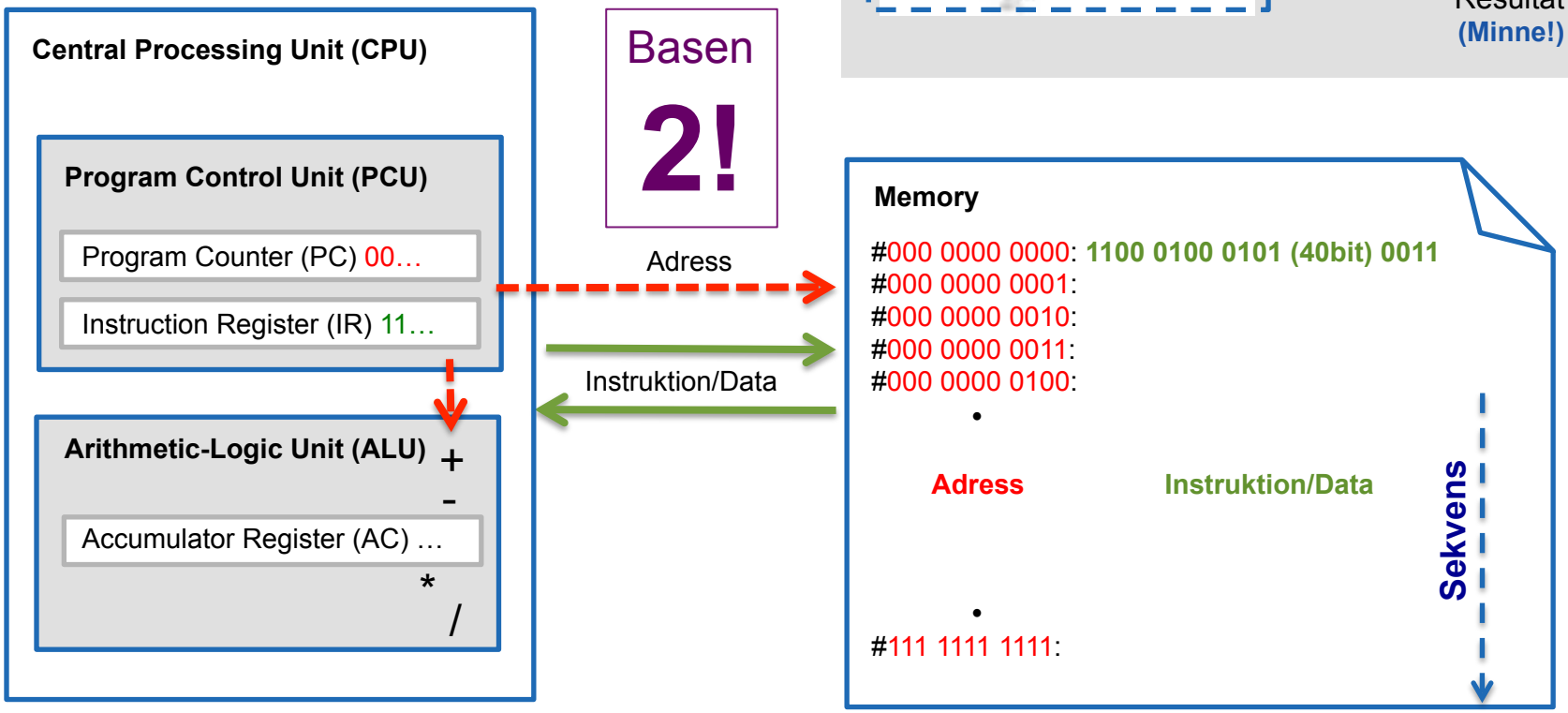
Hårdvara – vad kan en dator egentligen göra?!
,och hur?!

HW



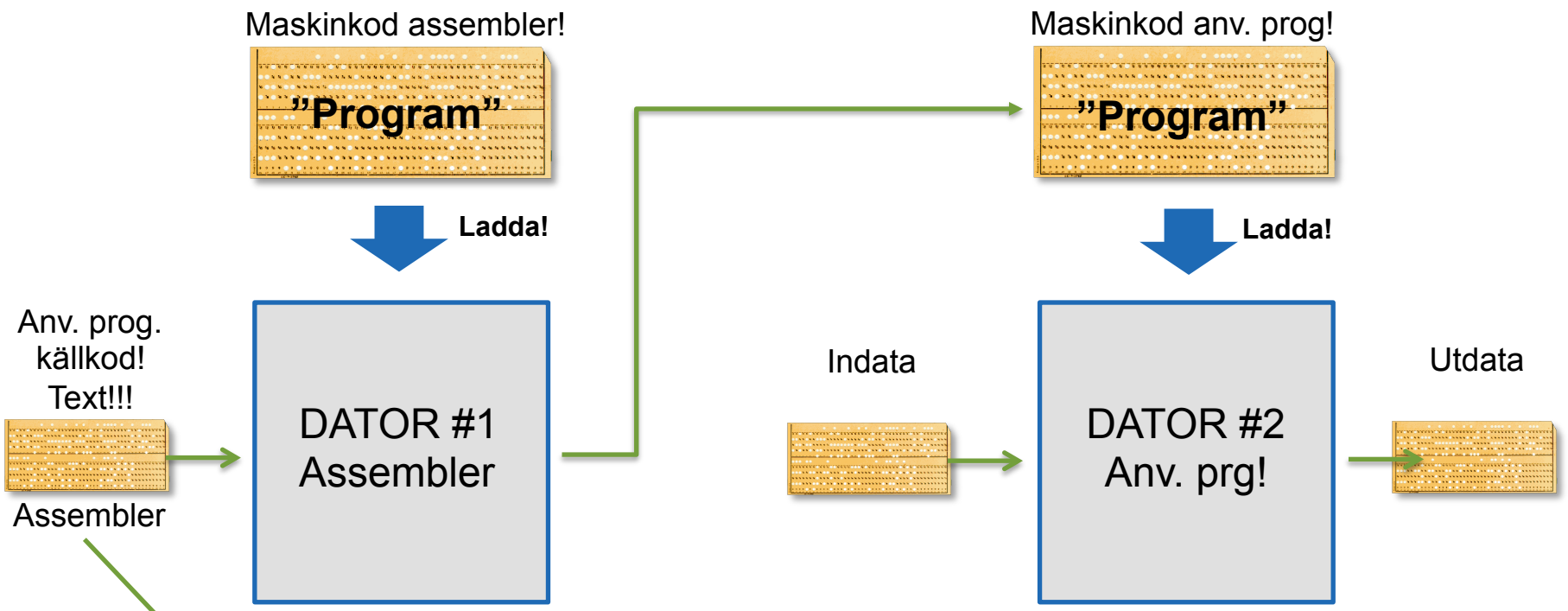
Kommer du ihåg?!





CPU, ALU, PC, Adress, basen 2/16, IR, AC, Sekvens

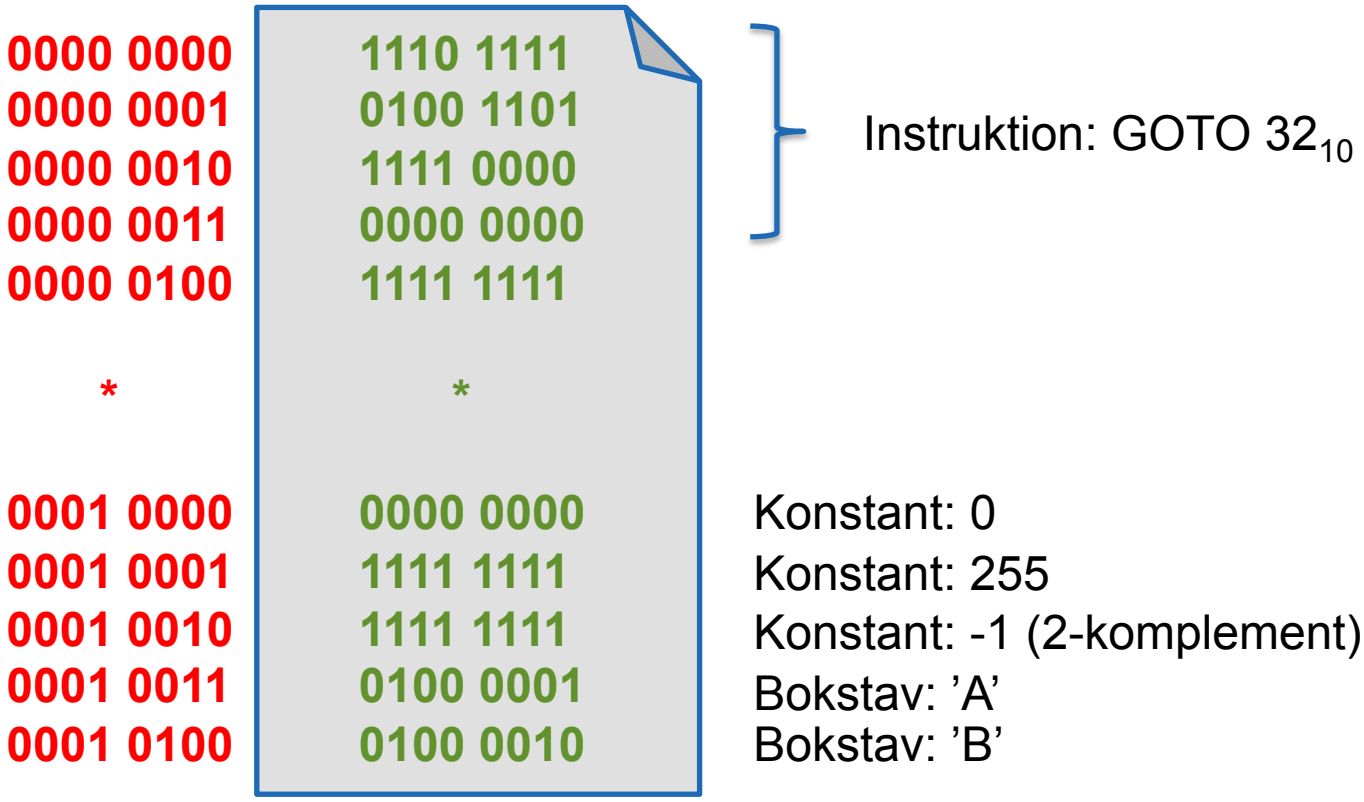
Symbolic programming, assembler…



Alla maskininstruktioner döps till ett kort namn.
Alla typer av konstanter kan ges förklarande namn.

Utvik: RAM/ROM & basen 16

Hur vet vi vad det binära talet i en minnescell betyder?

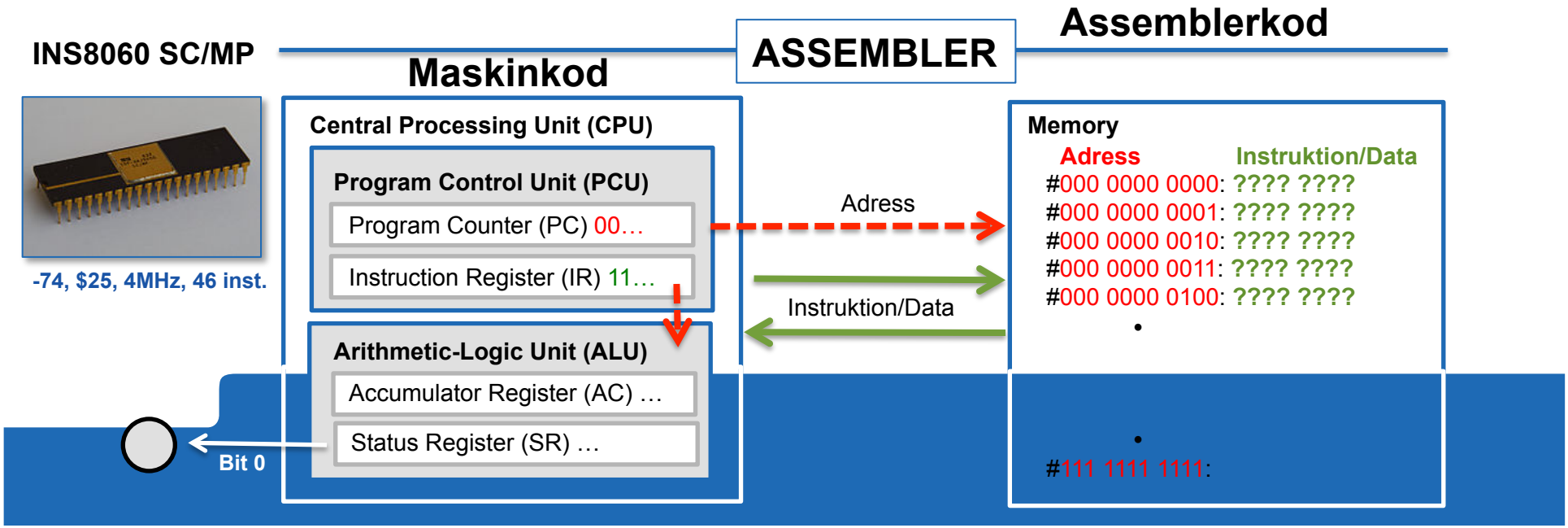


Instruktion: GOTO 32_{10}

Konstant: 0
Konstant: 255
Konstant: -1 (2-komplement)
Bokstav: ’A’
Bokstav: ’B’

Attans, det måste vi göra själva!
(Hur lagras ett stort tal?)

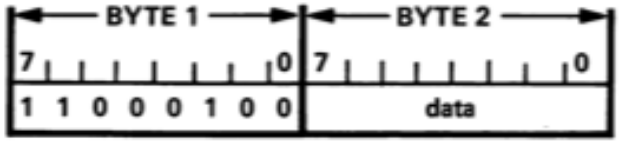
Järnkoll?!


INS8060 SC/MP
-74, $25, 4MHz, 46 inst.
Maskinkod
ASSEMBLER
Assemblerkod
Central Processing Unit (CPU)
Program Control Unit (PCU)
Program Counter (PC) 00…
Instruction Register (IR) 11…
Arithmetic-Logic Unit (ALU)
Accumulator Register (AC) …
Status Register (SR) …
Bit 0
Adress
Instruktion/Data
Memory
Adress
Instruktion/Data
#000 0000 0000: ???? ????
#000 0000 0001: ???? ????
#000 0000 0010: ???? ????
#000 0000 0011: ???? ????
#000 0000 0100: ???? ????
#111 1111 1111:

4 maskininstruktioner…

IMMEDIATE INSTRUCTIONS

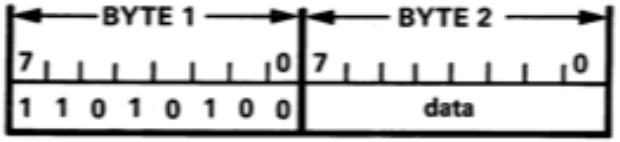
LOAD IMMEDIATE LDI(C4)



(AC)←(data)
The contents of the Accumulator (AC) are replaced by the data byte. The initial contents of AC are lost; the data byte is unaltered.

LDI	data	10

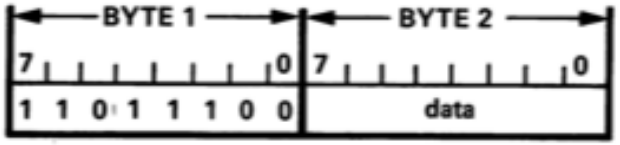
AND IMMEDIATE ANI(D4)



(AC)←(AC)∧(data)
The contents of the Accumulator (AC) are ANDed with the data byte, and the result is stored in AC. The initial contents of AC are lost; the data byte is unaltered.

ANI	data	10

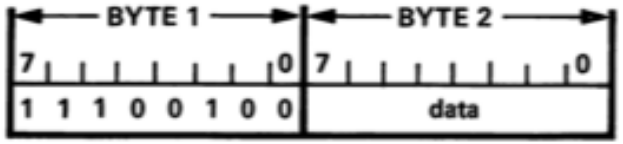
OR IMMEDIATE ORI(DC)



(AC)←(AC)∨(data)
The contents of the Accumulator (AC are inclusive-ORed with the data byte, and the result is stored in AC. The initial contents of AC are lost; the data byte is unaltered.

ORI	data	10

EXCLUSIVE-OR IMMEDIATE XRI(E4)



(AC)←(AC) ∇ (data)
The contents of the Accumulator (AC) are exclusive-ORed with the data byte, and the result is stored in AC. The initial contents of AC are lost; the data byte is unaltered.

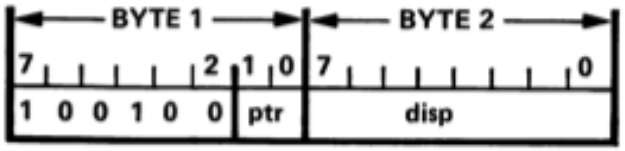
XRI	data	10

1100 0100 0000 0001 eller LDI 0x01

…och 4 till…

TRANSFER INSTRUCTIONS

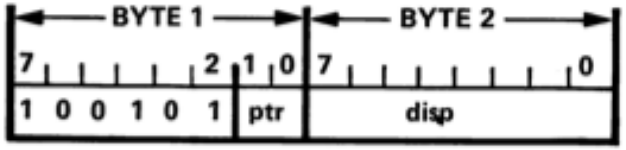
JUMP — JMP(90)



(PC) ← EA
The Effective Address (EA) replaces the contents of the Program Counter (PC). The next instruction is fetched from the location designated by the new contents of PC + 1.

JMP	disp (ptr)	11

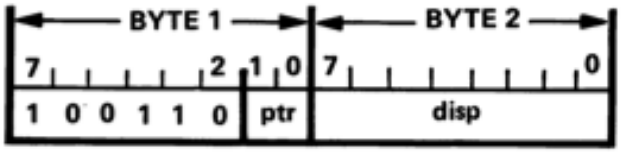
JUMP IF POSITIVE — JP(94)



IF (AC) ⩾ 0, (PC)←EA
If the contents of the Accumulator (AC) are positive or zero, the Effective Address (EA) replaces the contents of the Program Counter (PC). The next instruction is fetched from the location designated by the new contents of PC + 1.

JP	disp (ptr)	9 (no jump); 11 (jump)

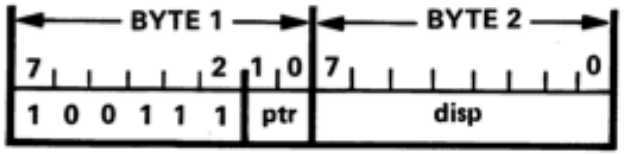
JUMP IF ZERO — JZ(98)



IF (AC) = 0, (PC)←EA
If the contents of the Accumulator (AC) are zero, the Effective Address (EA) replaces the contents of the Program Counter (PC). The next instruction is fetched from the location designated by the new contents of the PC + 1.

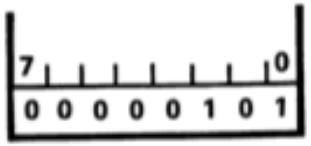
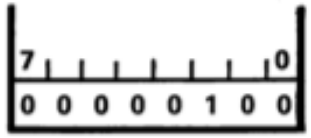
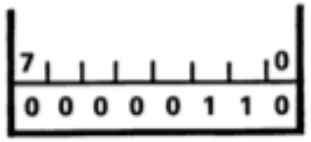
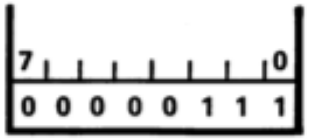
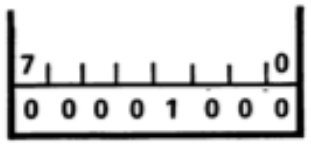
JZ	disp (ptr)	9 (no jump); 11 (jump)

JUMP IF NOT ZERO — JNZ(9C)



IF (AC) ≠ 0, (PC)←EA
If the contents of the Accumulator (AC) are not zero, the Effective Address (EA) replaces the contents of the Program Counter (PC). The next instruction is fetched from the location designated by the new contents of the PC + 1.

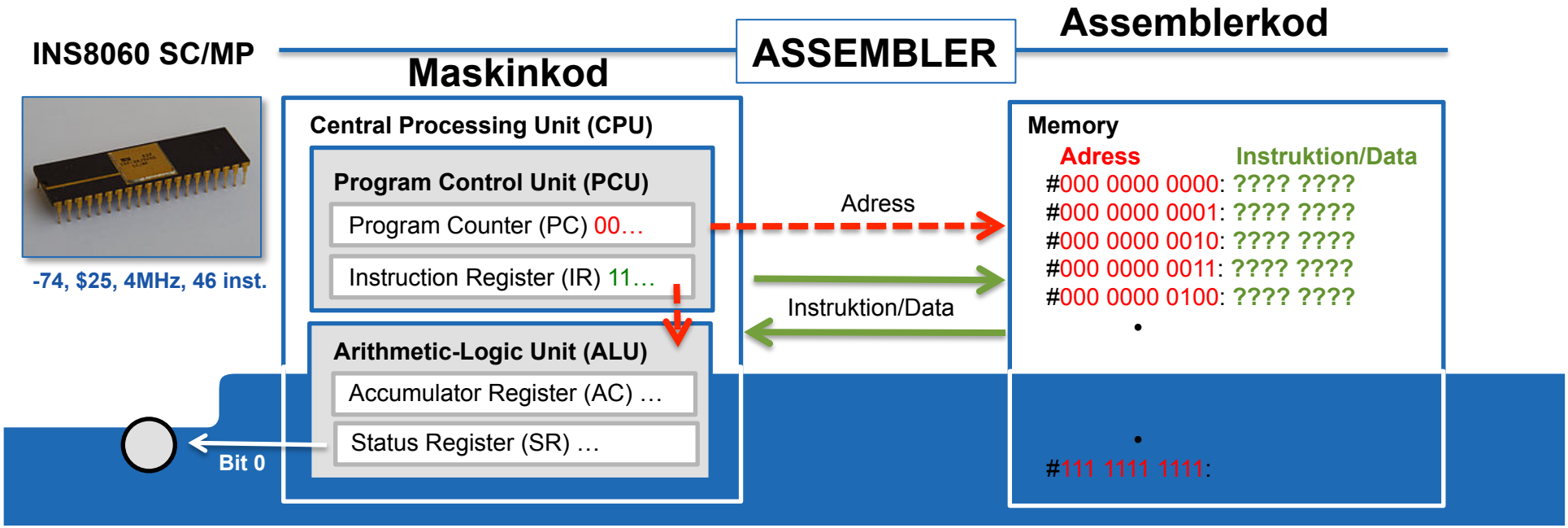
JNZ	disp (ptr)	9 (no jump); 11 (jump)

Instruction/Mnemonic		Operation/Description	Assembler Format	Execution Time (Microcycles)
ENABLE INTERRUPT 	**IEN(05)**	**(IE) ← 1** The Interrupt Enable (IE) Flag in the Status Register (SR) is set; the remaining bits in SR are not affected. The processor interrupt system is enabled. Interrupts will be processed as received *after* the next instruction is fetched and executed.	**IEN**	6
DISABLE INTERRUPTS 	**DINT(04)**	**(IE) ← 0** The Interrupt Enable (IE) Flag in the Status Register (SR) is cleared; the other bits in SR are not affected. The processor interrupt system is disabled. Interrupts which occur while the system is disabled will not be processed.	**DINT**	6
COPY STATUS TO AC 	**CSA(06)**	**(AC)←(SR)** The contents of the Accumulator (AC) are replaced by the contents of the Status Register (SR). The initial contents of AC are lost; the contents of SR are not altered.	**CSA**	5
COPY AC TO STATUS 	**CAS(07)**	**(SR)←(AC)** The contents of the Accumulator (AC) replace the contents of the Status Register (SR). SR bits 4 and 5 are external sense bits and are not affected by this instruction. The initial contents of SR (except for bits 4 and 5) are lost; the contents of AC are not altered. If IE is changed from 0 to 1 by this instruction, the interrupt system will be enabled *after* the next instruction is fetched and executed.	**CAS**	6
NO OPERATION 	**NOP(08)**	**(PC) ← (PC) + 1** The Program Counter (PC) is incremented by 1. The NOP instruction takes the minimum 5-microcycle execution time. Undefined opcodes encountered are considered to be one-byte or two-byte NOPs and may take 5 to 10 microcycles to execute, depending on the code.	**NOP**	5 (min) 10 (max)

Järnkoll?!

```
LDI 0x01
CAS
LDI 0x00
CAS
JMP -8
```

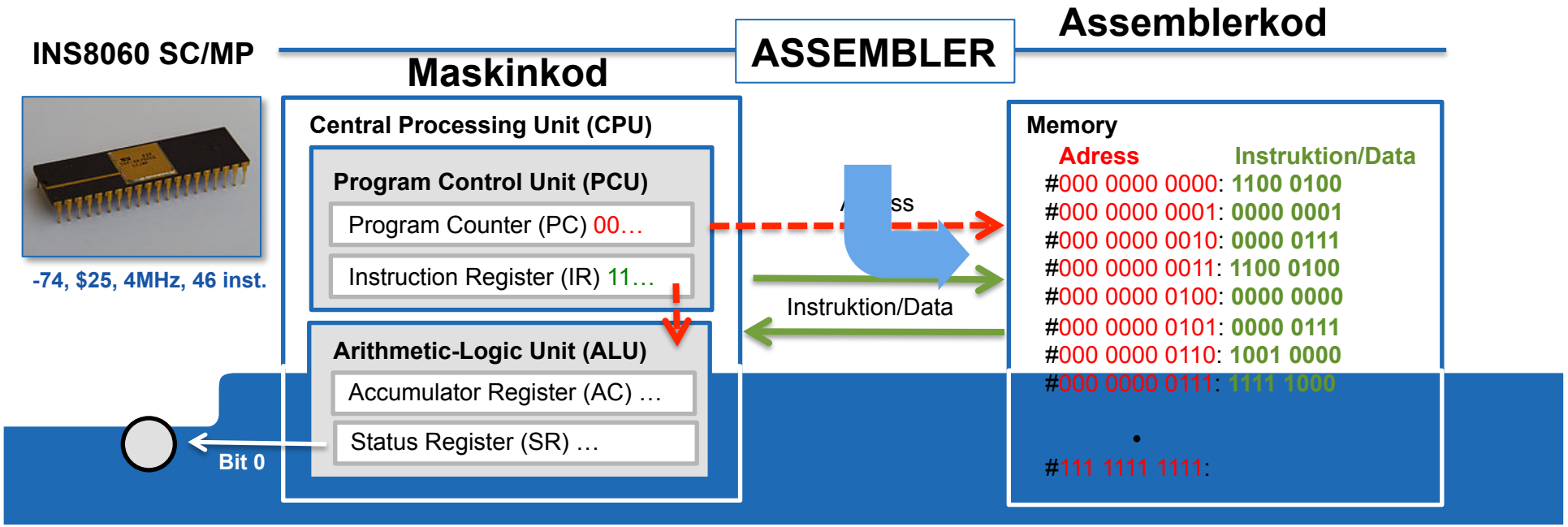
Blink.asm



Järnkoll?!

```
LDI 0x01
CAS
LDI 0x00
CAS
JMP -8
```

Blink.asm

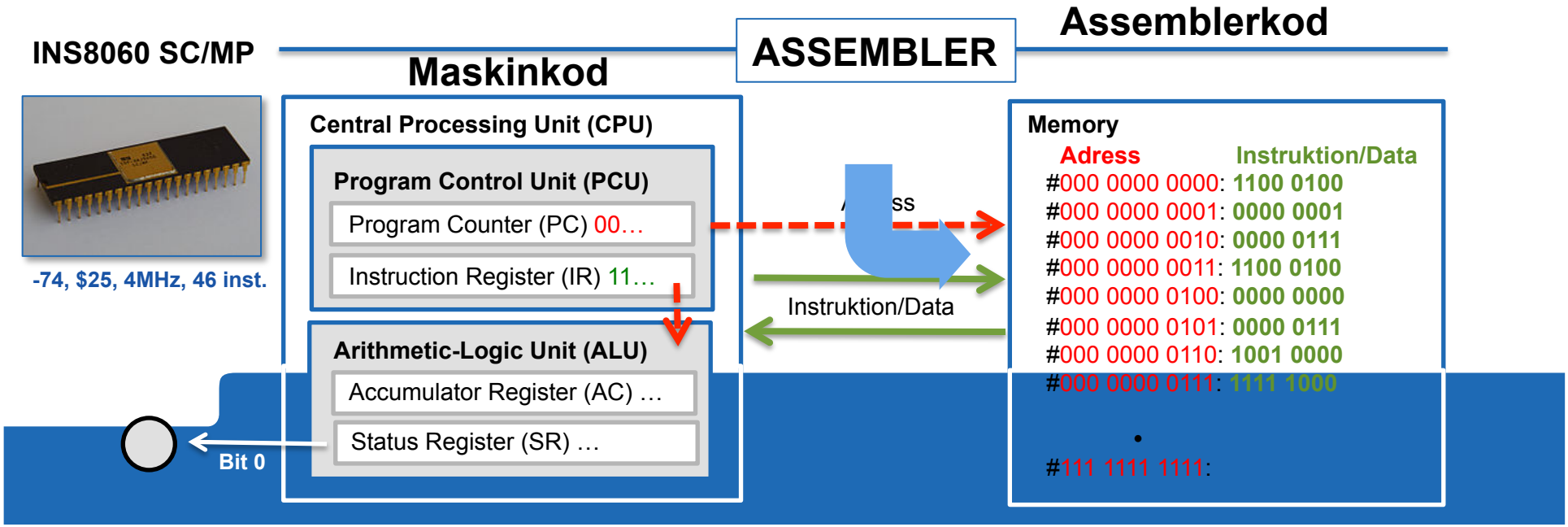


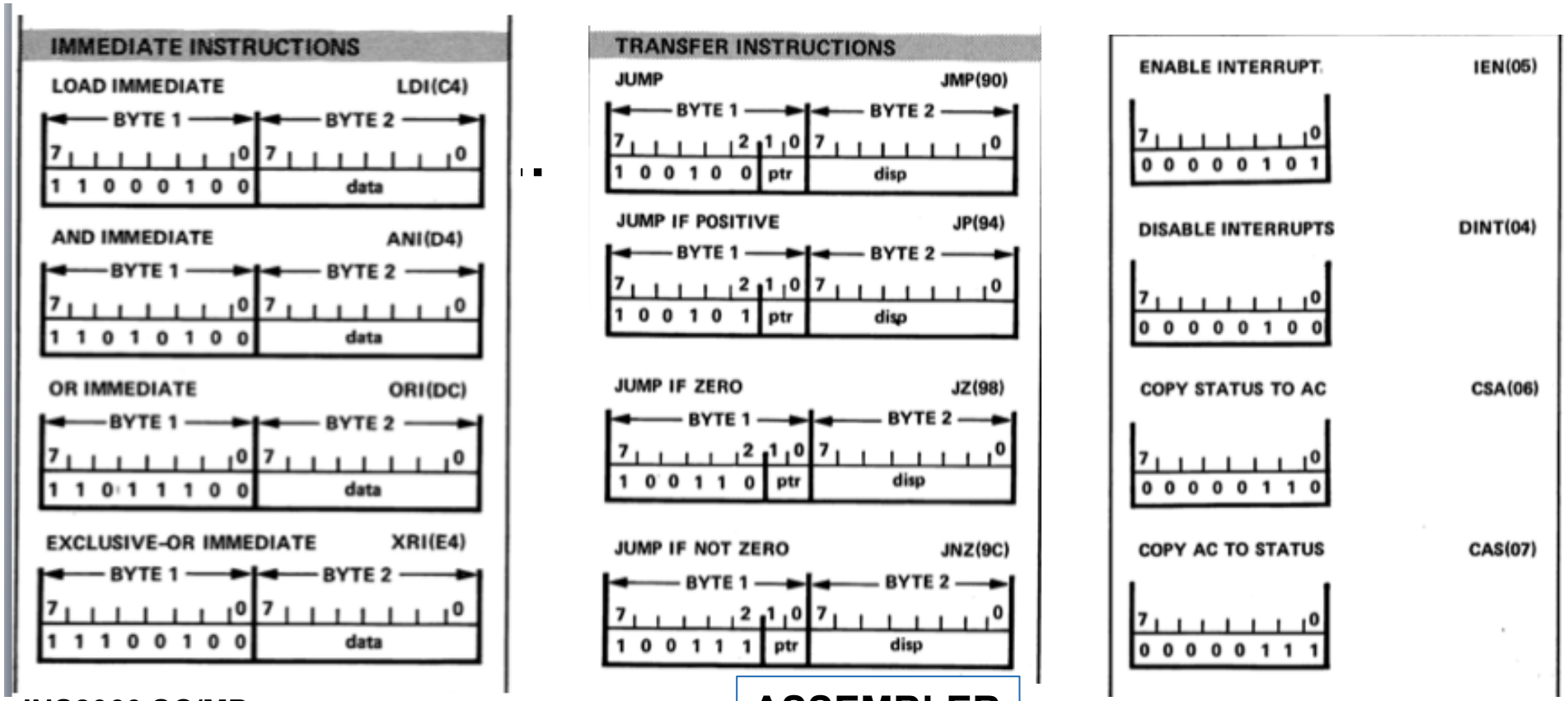
Järnkoll?!

```
LDI 0x01
CAS
LDI 0x00
CAS
JMP -8
```

Blink.asm

PC
IR
AC
SR
MEM





INS8060 SC/MP

Maskinkod

ASSEMBLER

-74, $25, 4MHz, 46 inst.

Central Processing Unit (CPU)

Program Control Unit (PCU)

Program Counter (PC) 00…

Instruction Register (IR) 11…

Arithmetic-Logic Unit (ALU)

Accumulator Register (AC) …

Status Register (SR) …

Bit 0

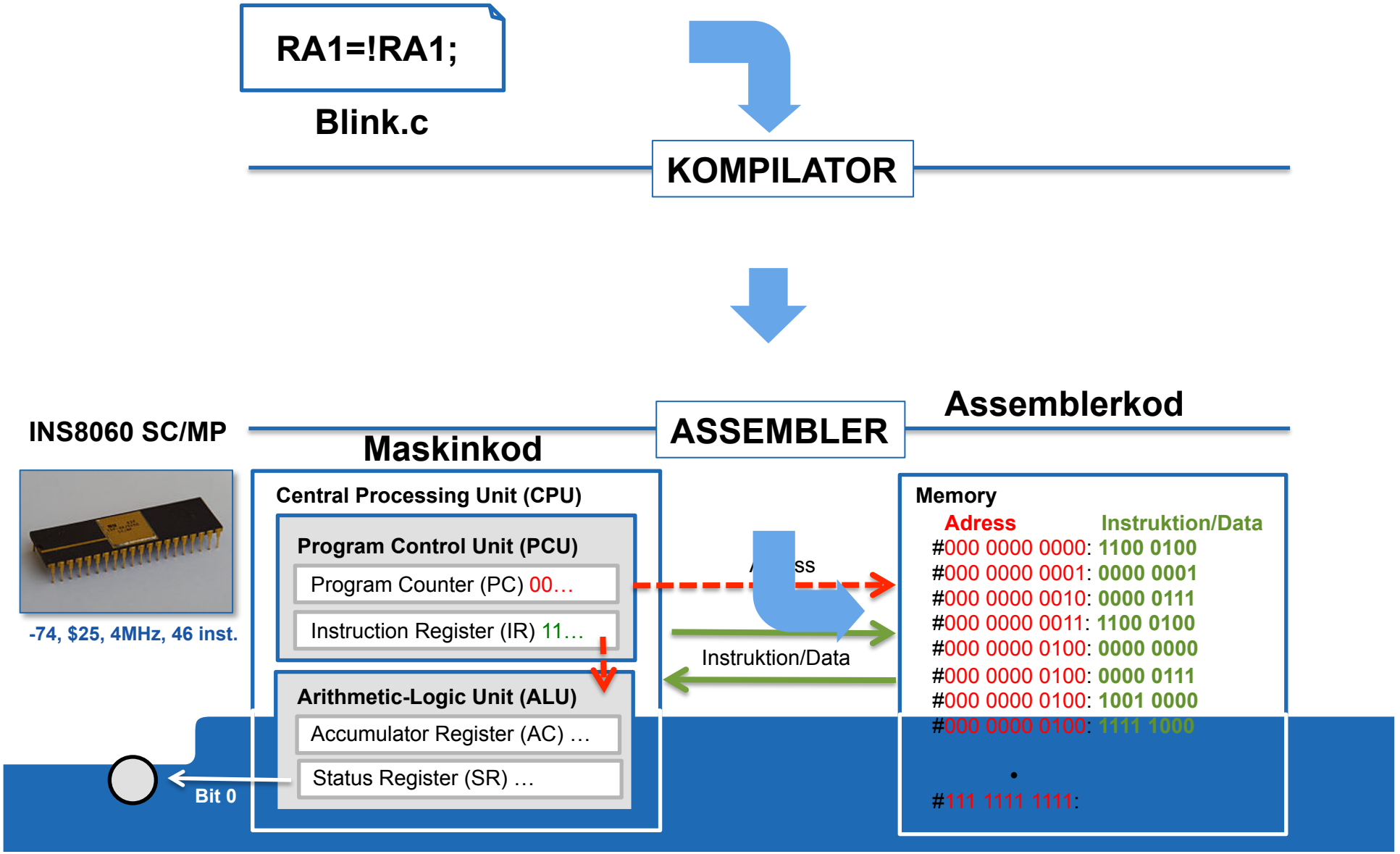
ss

Instruktion/Data

Memory

Adress	Instruktion/Data
#000 0000 0000:	1100 0100
#000 0000 0001:	0000 0001
#000 0000 0010:	0000 0111
#000 0000 0011:	1100 0100
#000 0000 0100:	0000 0000
#000 0000 0101:	0000 0111
#000 0000 0110:	1001 0000
#000 0000 0111:	1111 1000
•	
#111 1111 1111:	

Järnkoll?!


RA1=!RA1;
Blink.c
KOMPILATOR
Assemblerkod
INS8060 SC/MP
ASSEMBLER
Maskinkod
-74, $25, 4MHz, 46 inst.
Central Processing Unit (CPU)
Program Control Unit (PCU)
Program Counter (PC) 00…
Instruction Register (IR) 11…
Arithmetic-Logic Unit (ALU)
Accumulator Register (AC) …
Status Register (SR) …
Bit 0
Instruktion/Data
Memory
Adress
Instruktion/Data
#000 0000 0000: 1100 0100
#000 0000 0001: 0000 0001
#000 0000 0010: 0000 0111
#000 0000 0011: 1100 0100
#000 0000 0100: 0000 0000
#000 0000 0100: 0000 0111
#000 0000 0100: 1001 0000
#000 0000 0100: 1111 1000
#111 1111 1111:

Introduktionskurs i Datateknik

F1: Vad är en dator?

F2: programmeringsspråk?

L1: Hour of code, alla kan programmera, Diplom!

L2: Visuell programmering av svårare problem!

F3: Webbprogrammering – JavaScript!

L3: IDE, Web sida & JavaScript!

HTML

Kursmål

Sidor märkta med den här symbolen innehåller begrepp som är extra viktiga för dina fortsatta studier!

Varför finns det speciella programmeringsspråk?

Syntax och semantik!

Likheter: Datatyper, Flödeskontroll, Ofullkomligheter & Stora program.

Variabel är en namngiven plats i minnet för en viss typ av information!

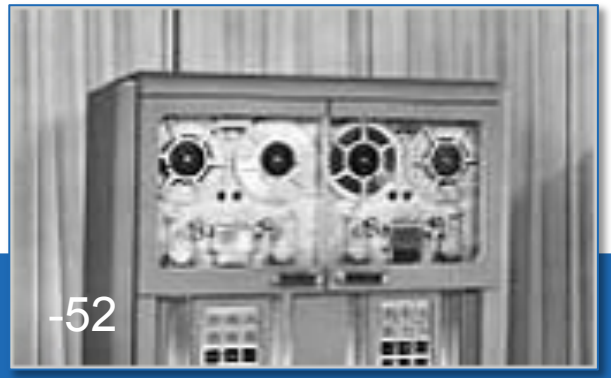
Flödeskontroll: Sekvens, Val, Upprepning och Funktioner.

Den andra generationen: 1955-1964

Transistorrevolutionen…

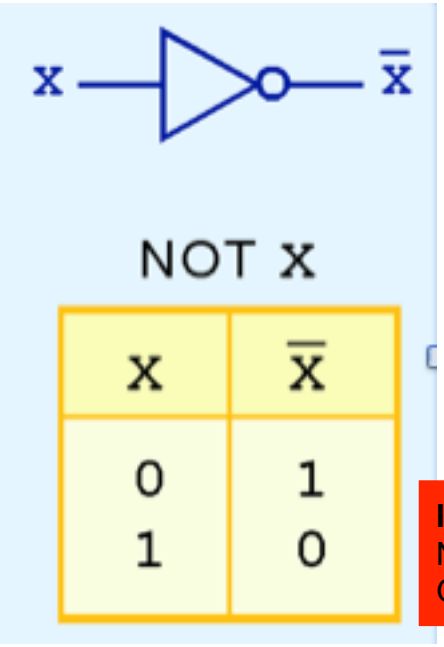
John Bardeen, William Shockley & Walter Brattain at Bell Labs, 1948

"There were 80 characters per card, and a good speed was 100 cards per minute." That's 133 characters per second. The first marketed tape drive from IBM, the 726, operated at 7,500 characters per second -- 56 times faster than the punch card rate.

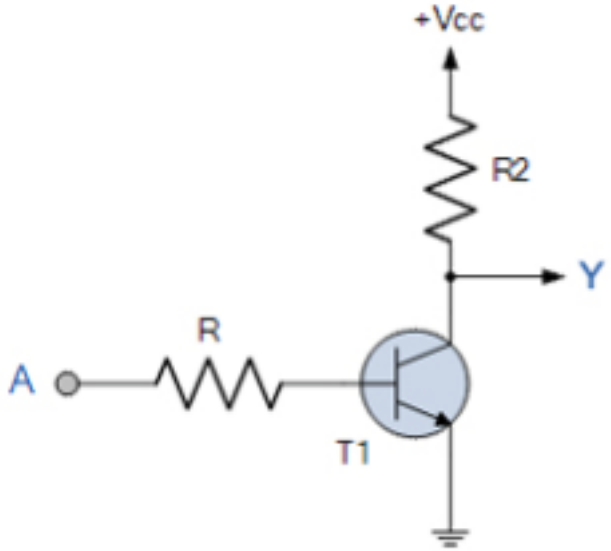


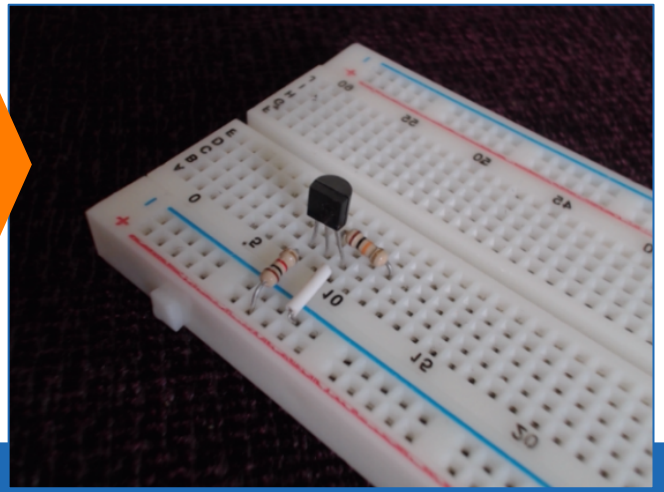
Den andra generationen: 1955-1964

Transistorrevolutionen…



IBM NOT Gate





Den andra generationen: 1955-1964

Transistorrevolutionen…

System performance	IBM -701 (-53)
Electrostatic storage capacity:	20,480 digits.
Magnetic drum capacity:	81,920 digits.
Magnetic tape capacity:	More than 8 million digits without changing tape.
Addition and subtraction:	More than 16,000 operations per second.
Multiplication and division:	More than 2,000 operations per second.
Tape reading and writing speed:	12,500 digits per second.
Drum reading and writing speed:	8,000 digits per second.

Batch = Ett enskilt jobb.

#1: BATCH MONITOR – BÖRJAN TILL ETT OPERATIVSYSTEM (OS)

Monitor = Styrprogram som alltid fanns i datorn!

#2: FORTRAN, COBOL – KOMPILATOR (Maskinoberoende)

-57 -59

Den andra generationen: 1955-1964

Maskinkod, Assembler, Högnivåspråk…

```
#0000 0000 0000: 1100 0100
#0000 0000 0001: 0000 0001
#0000 0000 0010: 0000 0111
#0000 0000 0011: 1100 0100
#0000 0000 0100: 0000 0000
#0000 0000 0101: 0000 0111
#0000 0000 0110: 1001 0000
#0000 0000 0111: 1111 1000
```

MASKINKOD!
(Processorspecifik)

Assembler

```
POR     LDI 0x01    //Turn lamp…
        CAS         //          …on!
        LDI 0x00    //Turn lamp…
        CAS         //          …off!
        JMP POR     //Forever!
```

ASSEMBLERKOD!
(Processorspecifik)

Kompilator

(Interpretator)

(Mathlab)

(Kompilerar en rad i taget allt eftersom pro-grammet körs)

```
Int main(void) {

  while (true)          //Forever…
    RA0++;              //…toggle RA0!

}
```

C-KOD!
(Generell)

Fördjupning: Programmeringsspråk.

Varför har vi ett programmeringsspråk?

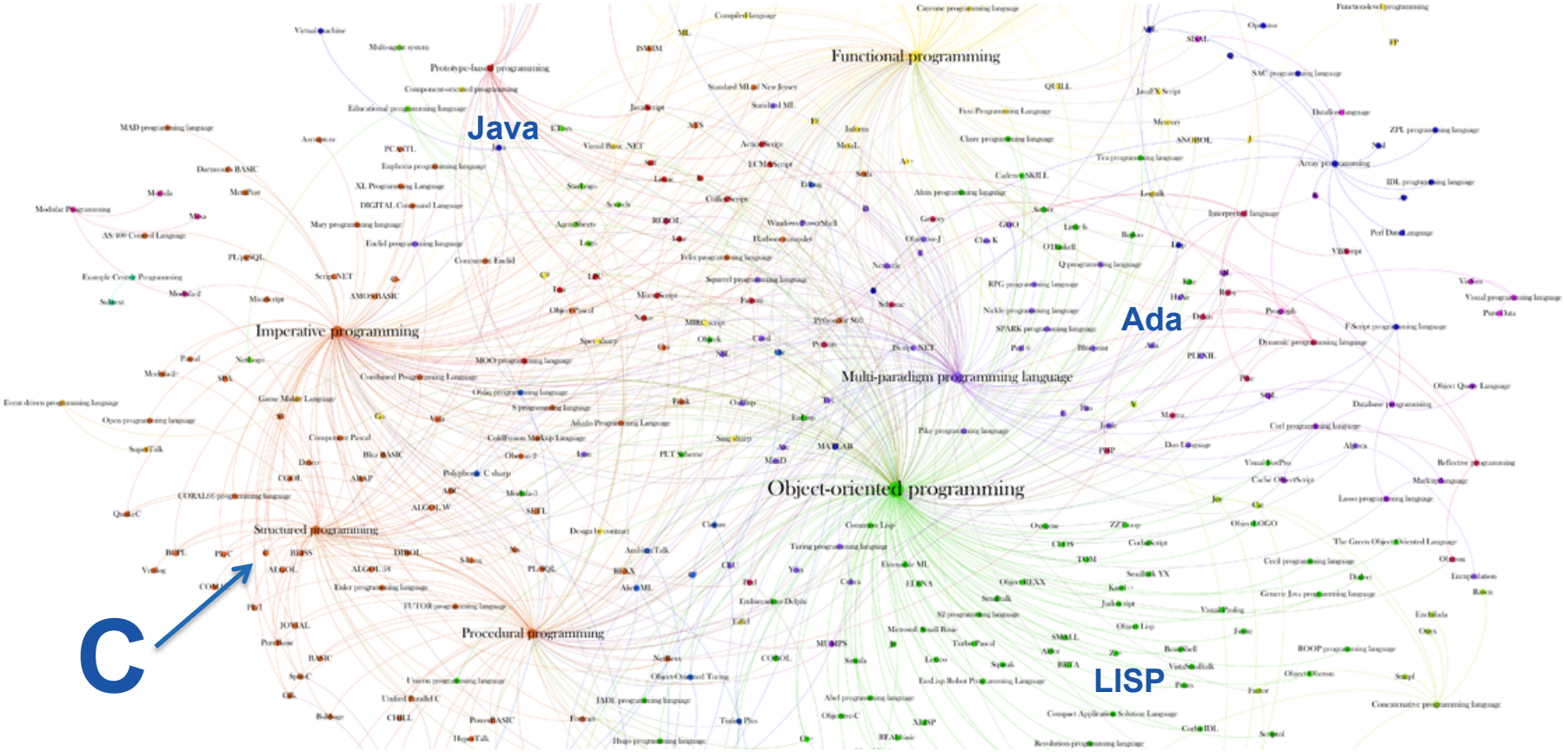
I cdnuolt blveiee taht I cluod aulaclty uesdnatnrd waht I was rdanieg the phaonmneal pweor of the hmuan mnid! Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer inwaht oredr the ltteers in a wrod are, the olny iprmoatnt tihng is taht the frist and lsat ltteer be in the rghit pclae. The rset can be a taotl mses and you can sitll raed it wouthit a porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe. Amzanig huh? Yaeh, and I awlyas thought slpeling was ipmorantt.

Det är för svårt att skapa en översättare!

(=Kompilera, översätta till maskinkod!)

Fördjupning: Programmeringsspråk.

Det finns 1000-tals programmeringsspråk…



Generell, maskinnära, processorienterad, funktionell…

Fördjupning: Programmeringsspråk.

Syntax, semantik och reserverade ord!

```
while (   closed  (  'A'  ,  number  +  1  )   ) ;
```

”A **programming language** is a **formal notation** for describing algorithms to be executed by a computer. Like all formal notations, a programming language has two components: **syntax** and **semantics**”

begränsat

Formellt språk som inte kan misstolkas!

(= Lätt att översätta!)

Fördjupning: Programmeringsspråk.

Syntax, semantik och reserverade ord!

while (closed ('A' , number + 1)) ;

Vad är ”ord” (tokens) i språket!

Syntax

Literals **Operators**

Identifiers
*[A-Za-z_][A-Za-z0-9_]**

(Tokens)

Punctuation marks
,;(){}:;white space

Semantik

Vad är en mening i språket!

(**Keywords ≈50!**)

Antalet mellanslag spelar nästan alltid ingen roll!

Likheter…

Möjligheter att mappa verkliga storheter till datorvariabler

DATA TYPES

Syntax

CTRL STRUCT

Möjligheter att påverka sekvensen av instruktioner!

Literals Operators

Identifiers
*[A-Za-z_][A-Za-z0-9_]**

Tokens

Punctuation marks
,;(){}:;white space

(Keywords)

Egenheter att se upp med!

PRG CORR

Semantik

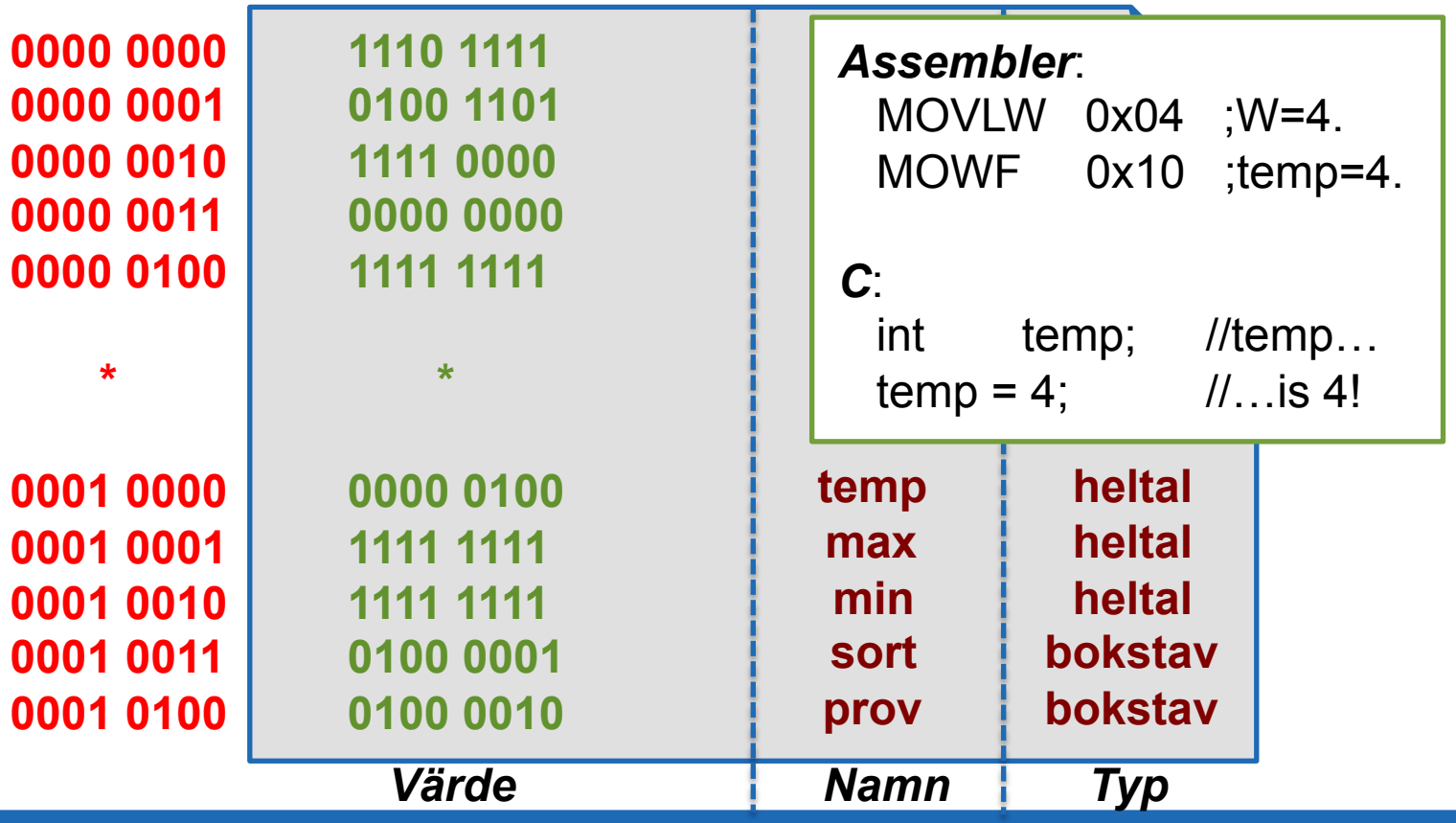
PRG IN LARGE

Möjligheter att skapa stora komplicerade program.

Vi börjar med att titta på vad data typer är…

Datatyper…

Med hjälp av en datatyp kan en platser i minnet namnges där ett värde av en speciell typ kan lagras – en variabel!



Ex c: int temp;

Maskinoberoende, Korrekt användning, Otydligheter utklarade & noggrannhet.

Fördjupning: Programmeringsspråk.

Sammanfattning datatyp!

Garbage Collection

Vilka värden….

-127..128
False
True

TYPE + * -

…och vilka operationer!

Binding Reference

VALUE

Constant uninitialized

Abstract

En variabel i ett programmeringsspråk används för att (mellan-)lagra ett värde.
En variabel har ett namn och 4 attribut!

Casting

Visible Side-effects

SCOPE

Static Dynamic

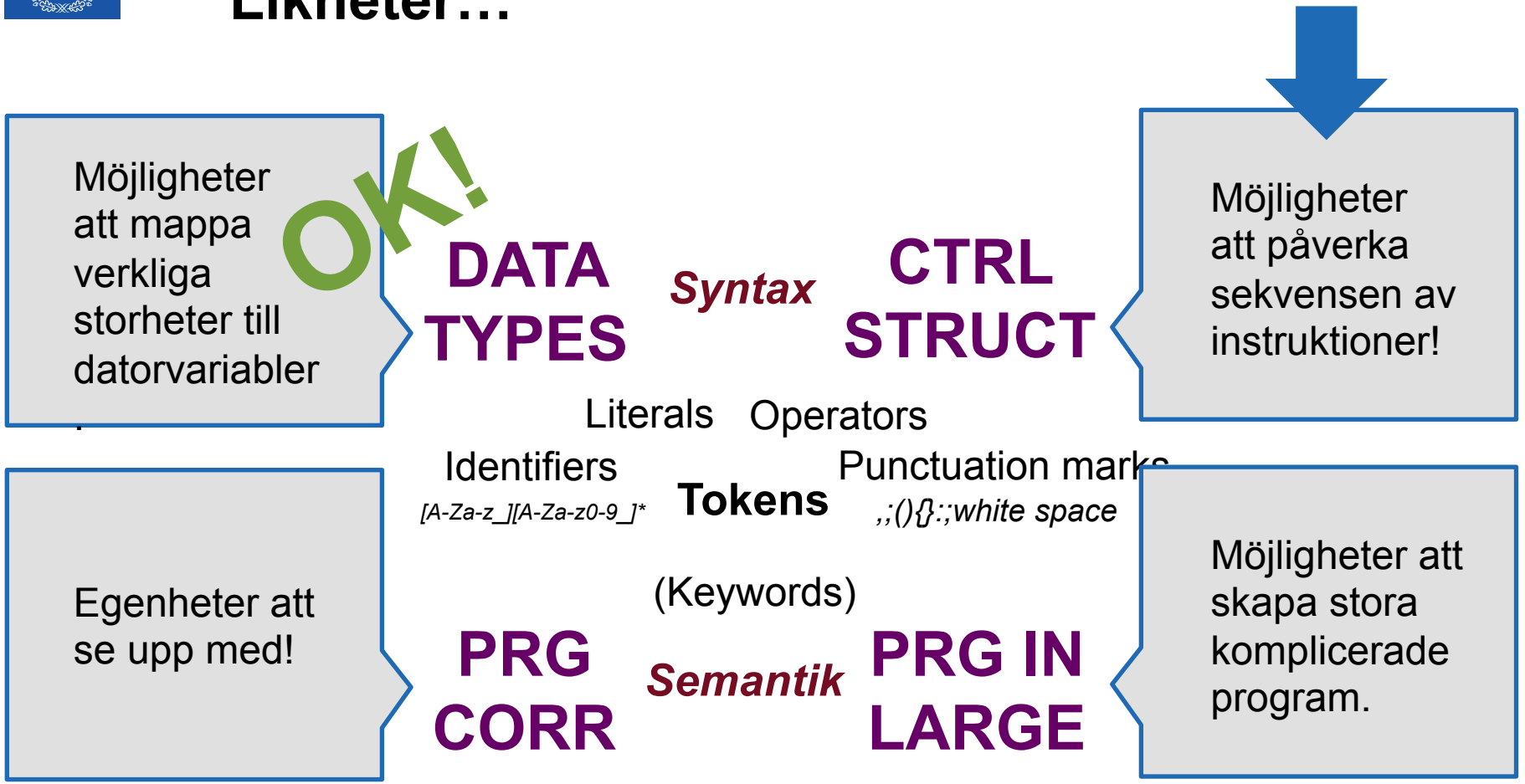
Allocation Automatic

LIFETIME

Static Dynamic

Gråmarkerade begrepp får vänta till C/Java-kurserna!

Likheter…



Vad är (flödes-)kontroll strukturer?

Fördjupning: Programmeringsspråk.

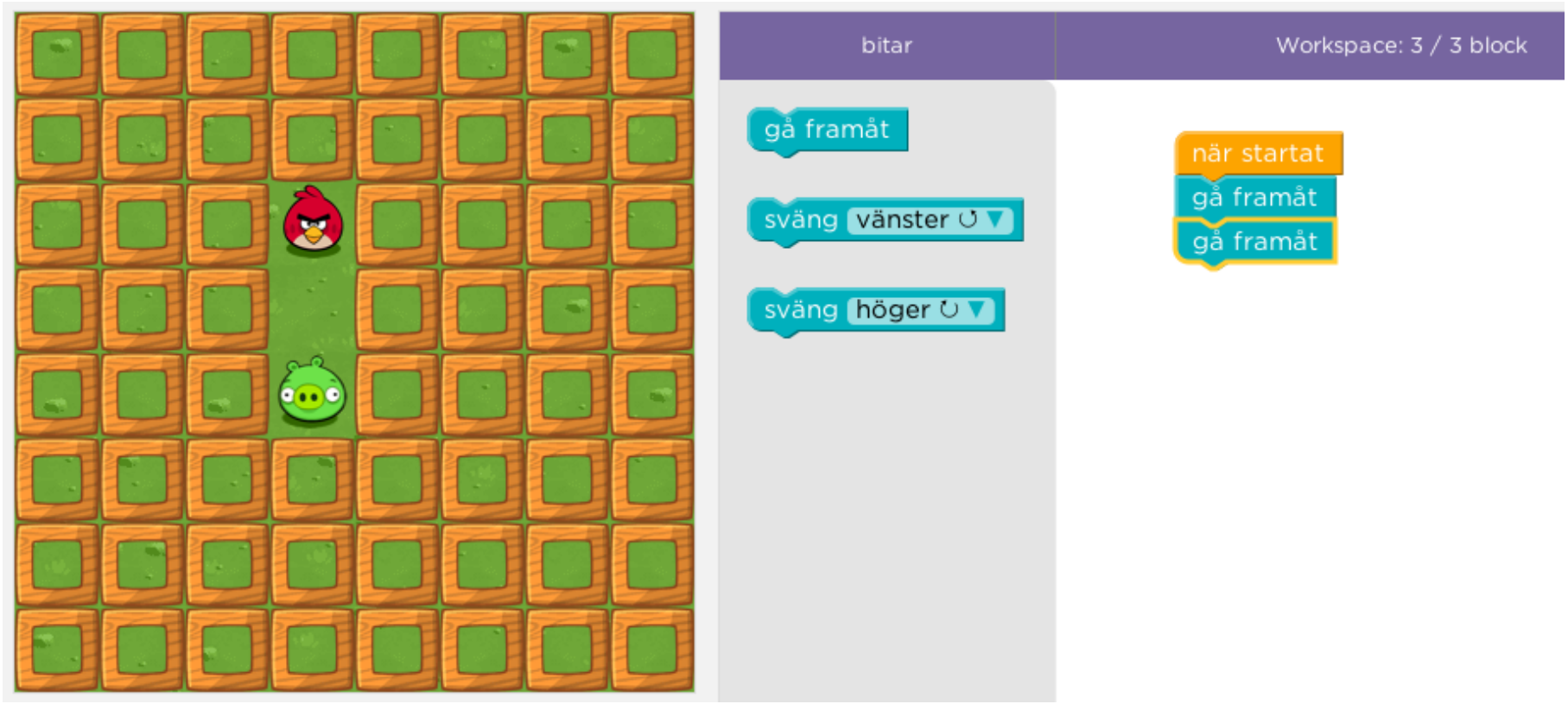
Control Structures

En dator utför alltid ”nästa rad”, om den inte får andra instruktioner!

Mät ugnstemperaturen.
Om ugnstemperaturen är över 205 slå av värmen.
Om ugnstemperaturen är under 195 slå på värmen.
Mät kött-temperaturen.
Om kött-temperaturen är under 65 börja om!
Stäng av värmen.

- **Statement-Level Control Structures**
  - Sequencing
  - Repetition
  - Selection
  - Jump
- **Unit-Level Control Structures**
  - Explicitly Called Units
  - Implicitly Called Units
  - Concurrent Units

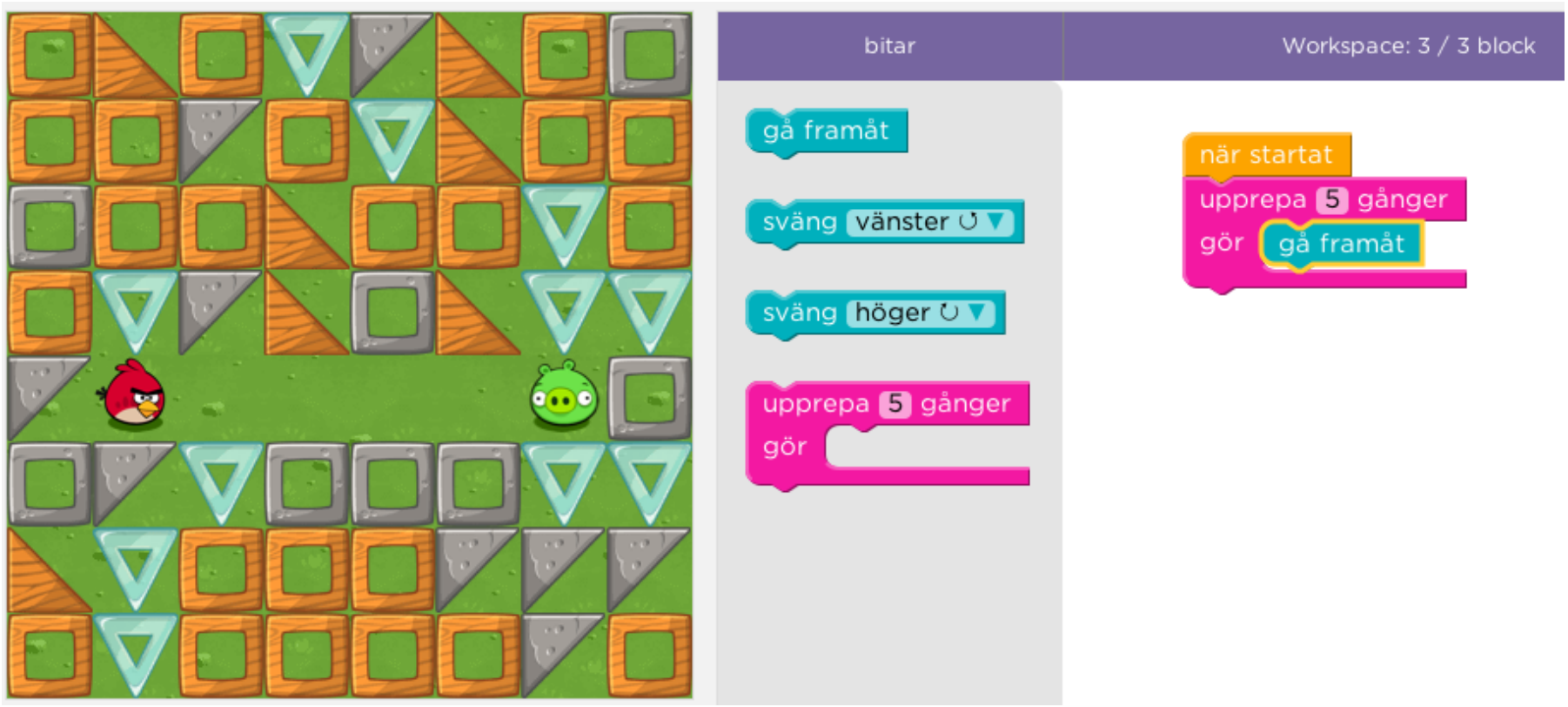
Sequencing



Som i ett recept: nästa rad, nästa rad, nästa rad, nästa rad…

Fördjupning: Programmeringsspråk.

Repetition (for, while, until, do)



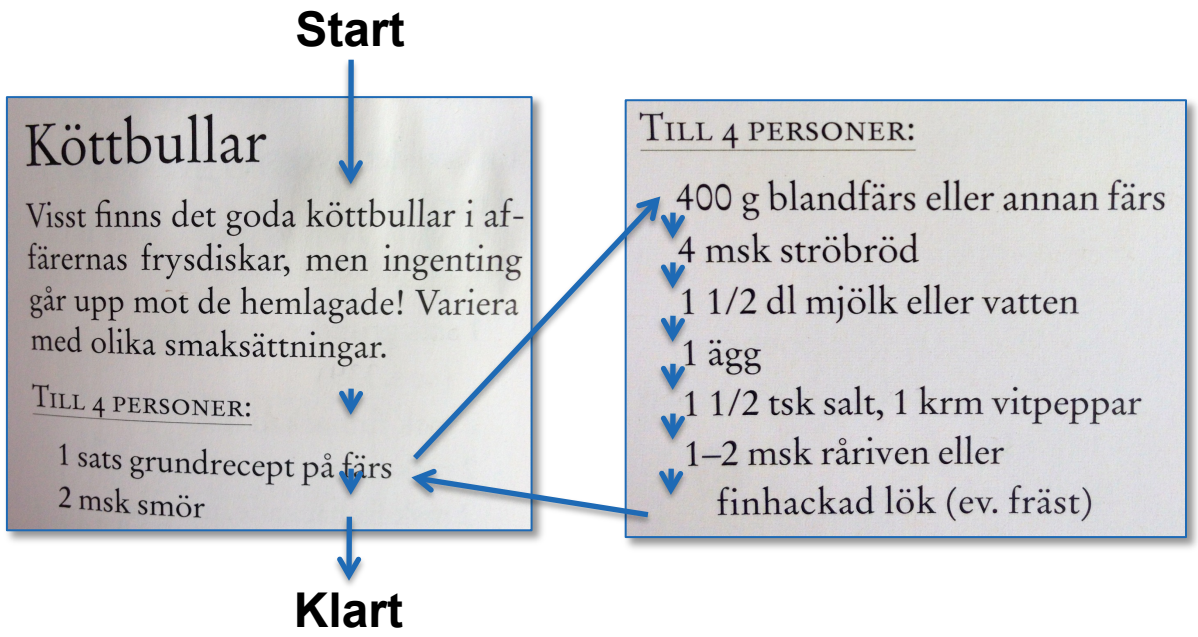
Som i ett recept: Tillsätt 5 äggulor, ett i taget…

Selection (if, then, else, switch, case, default, other(wise))

Som i ett recept: Om ”*russinbullar*” tillsätt 1 dl russin!

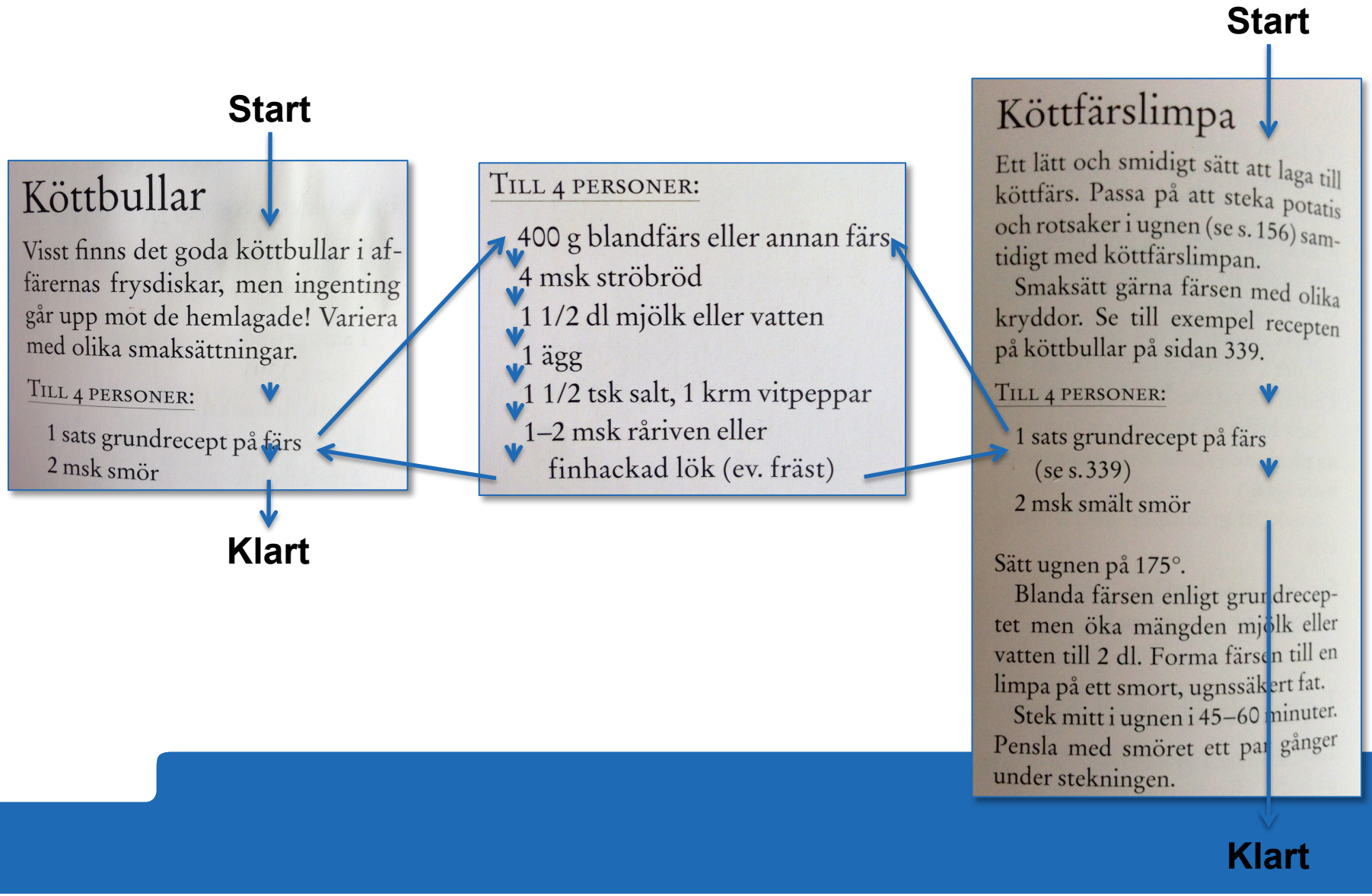
Fördjupning: Programmeringsspråk.

Explicitly Called Units (subroutines, function, procedure)



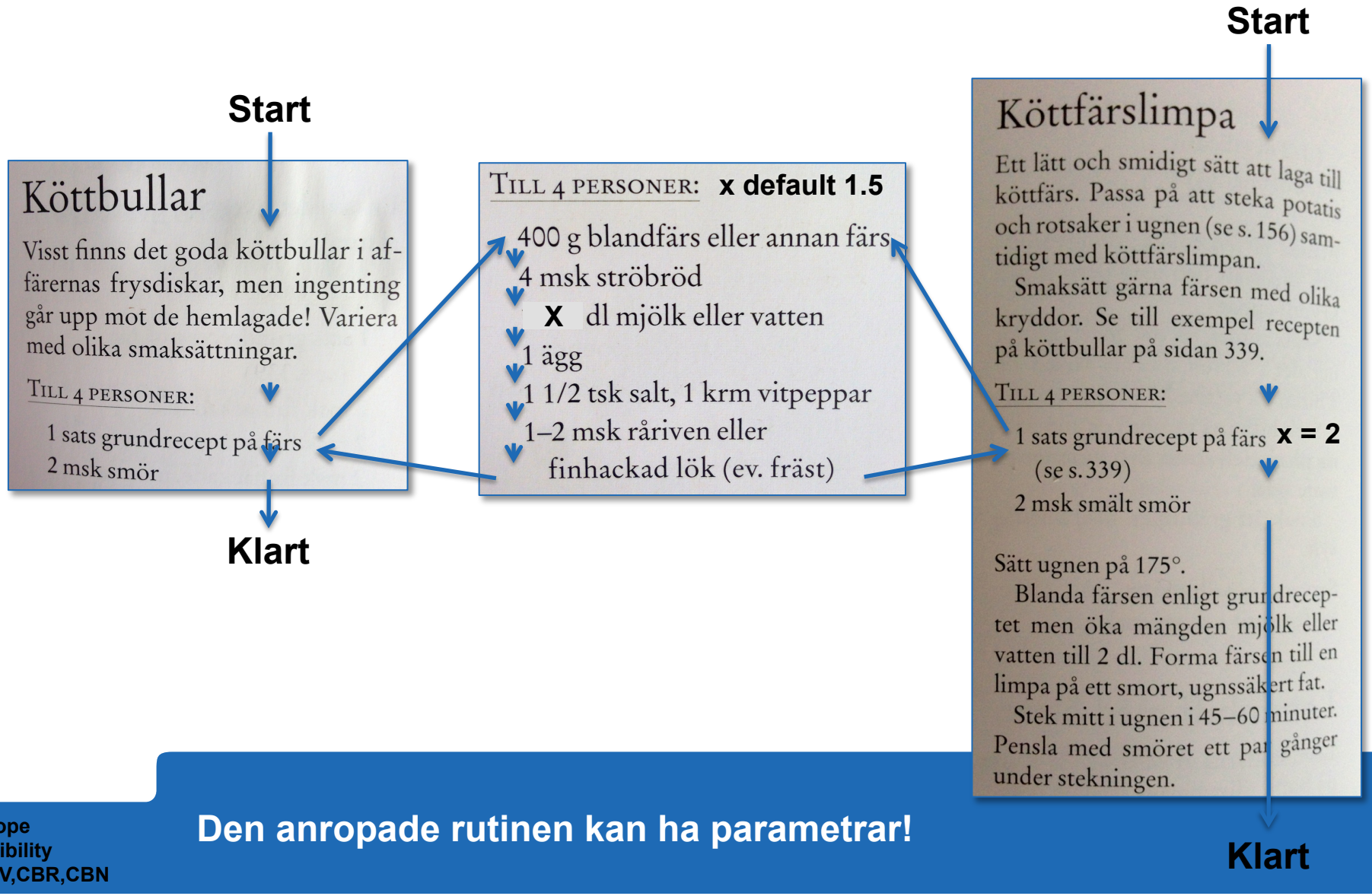
Fördjupning: Programmeringsspråk.

Explicitly Called Units (function, procedure)



Fördjupning: Programmeringsspråk.

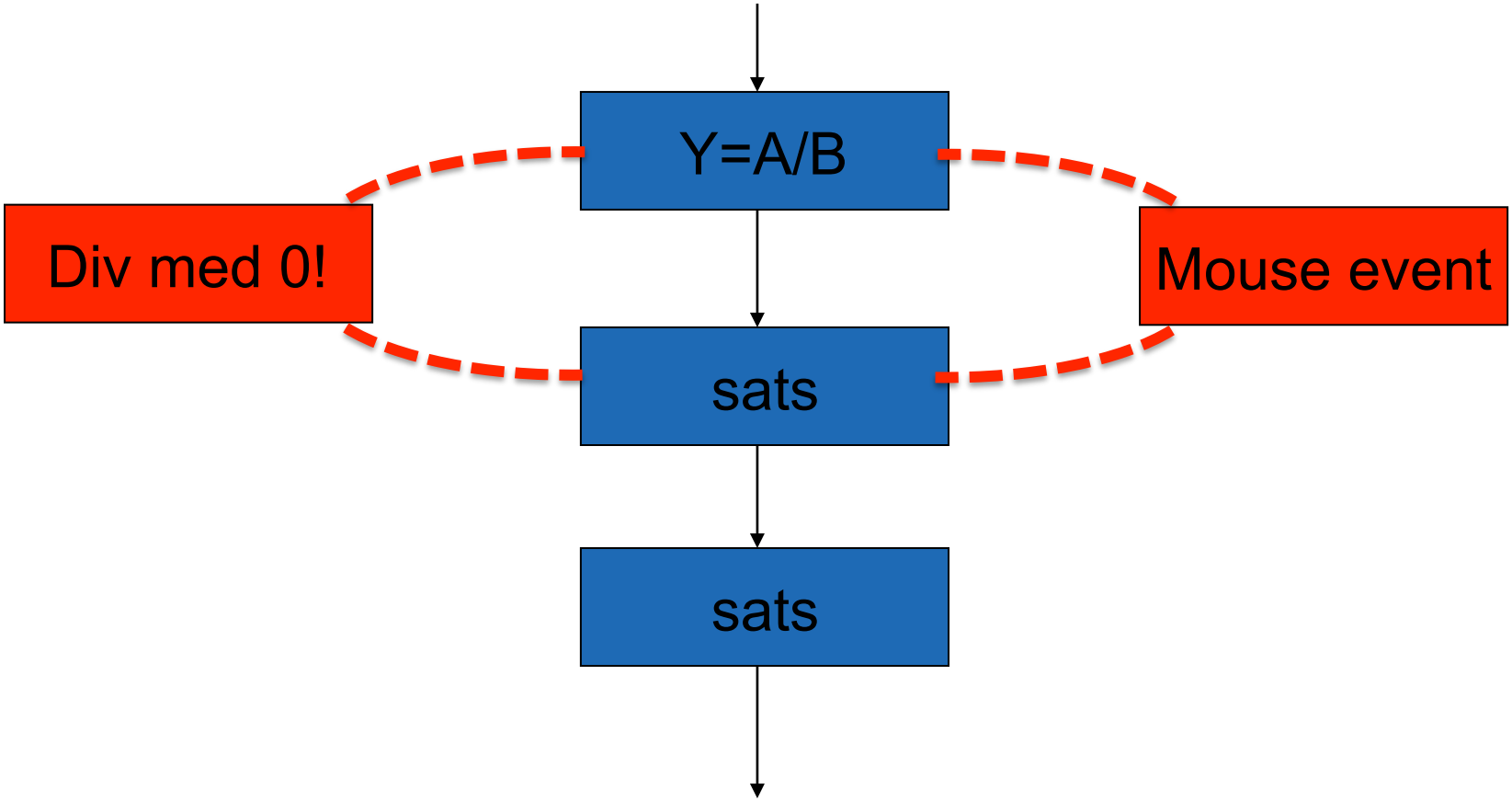
Explicitly Called Units (subroutines, function, procedure)



Scope
Visibility
CBV,CBR,CBN

Den anropade rutinen kan ha parametrar!

Implicitly Called Units (exceptions, interrupt)



Fördjupning: Programmeringsspråk.

Concurrent Units (task, fork, process, thread)

Resource starvation

Race conditions

Deadlock

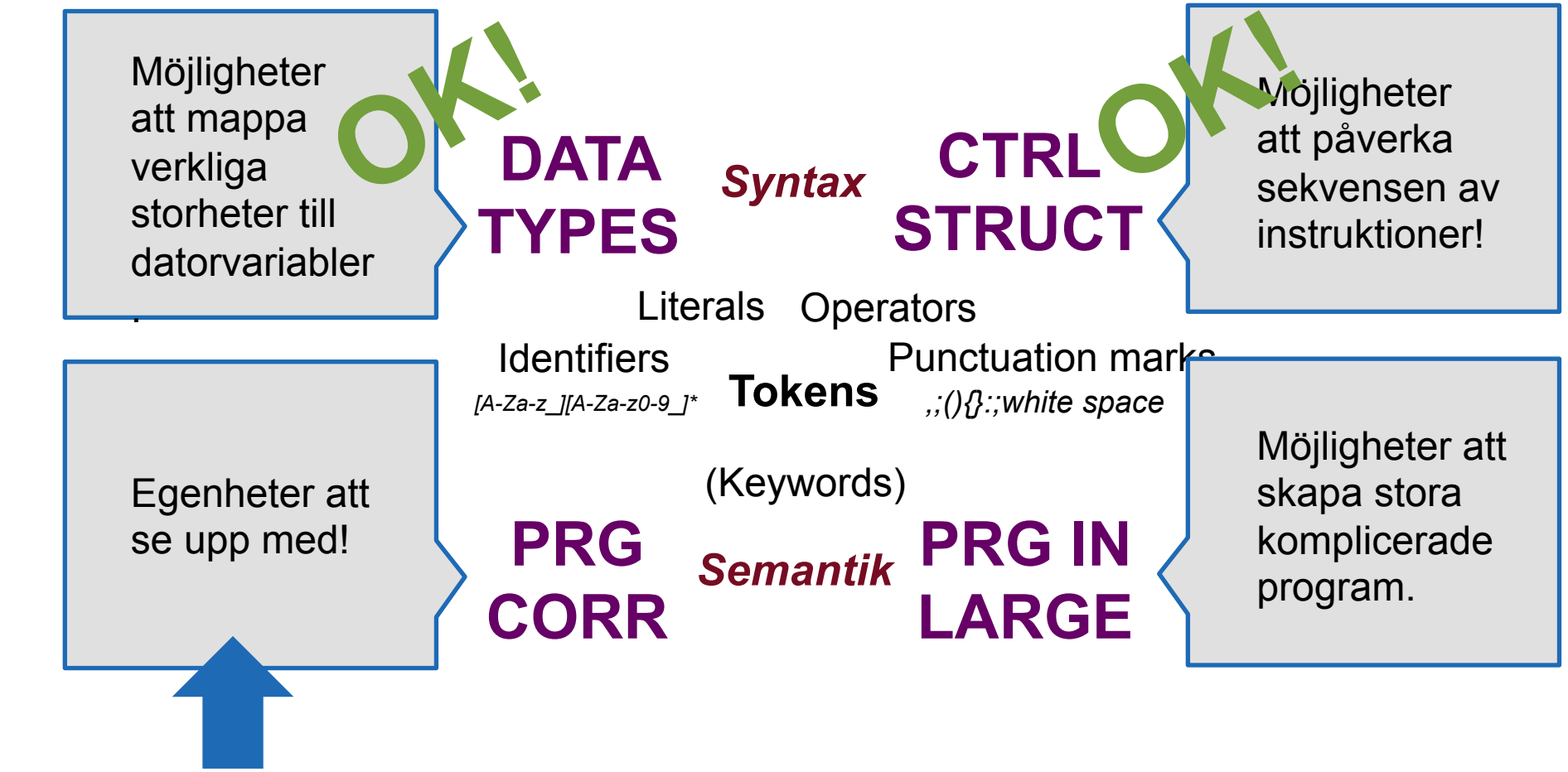
Producer → **Consumer**

Semaphores

Monitors

Mutex

Likheter…



Ofullkomligheter…

Correctnes…

Vad händer vid division av heltal?

Vad händer vid division med noll?

Vad händer om ett tal blir ”större” än vad variabeln kan hantera?

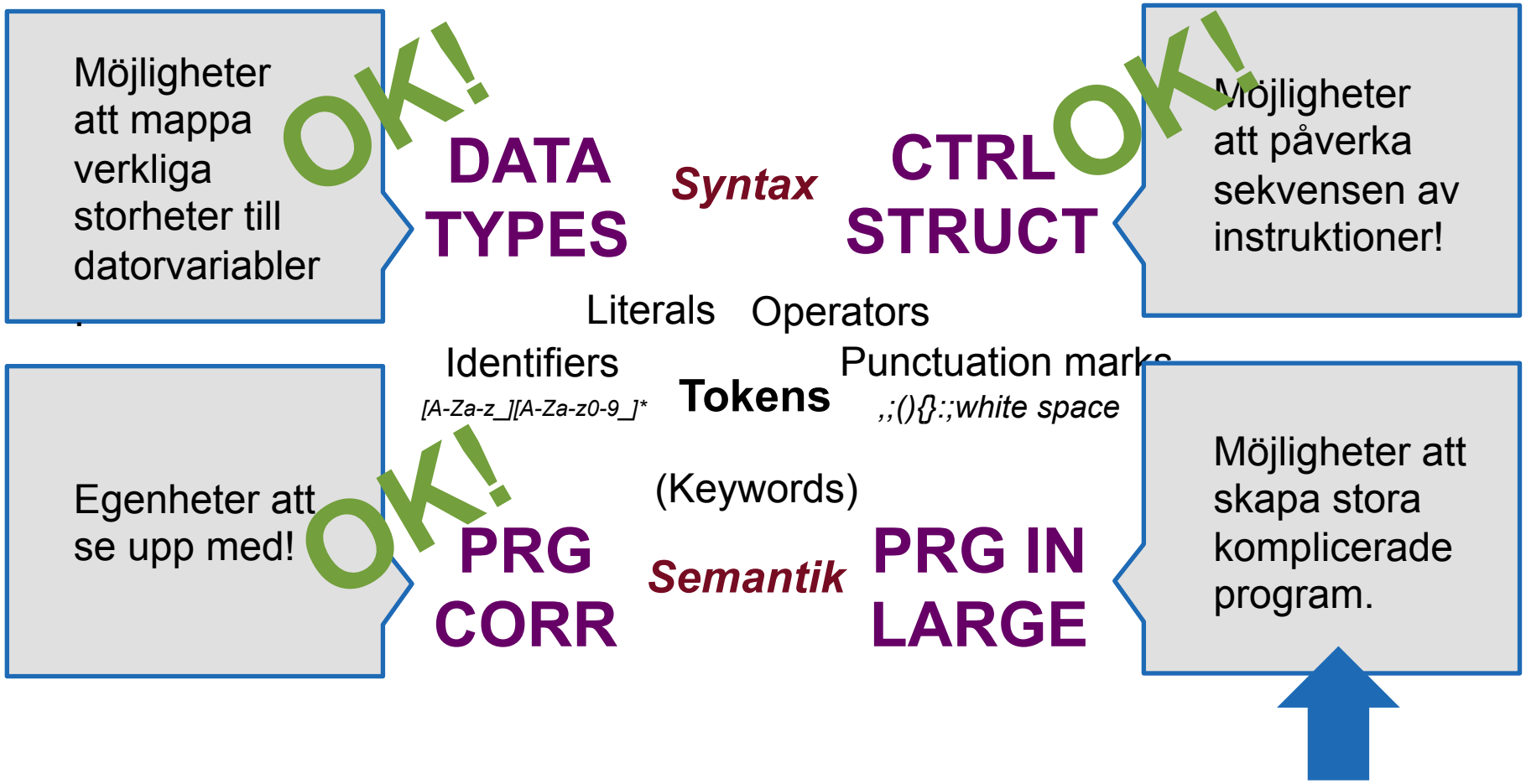
Vad händer om en bokstav adderas till ett tal?

Kan ord ”plussas” samman?

Vad händer när en variabel utan värde används?

Alla programmeringsspråk har minst någon märklig egenhet!

Likheter…



Vadå skapa stora program?

Programming In Large…

```c
#include <stdio.h>
int main(void) {

    printf("Hello World!");

}
```

”Hela programmet” återfinns i huvud-funktionen!

”Huvudprogrammet”, det som heter main(void){ … } tar hjälp av sorteringsfunktionen bubbla!

Ett ”stort” program kan ta hjälp av hundratals funktioner, där många kan vara generella och lika från program till program!

Språket borde ha stöd för det!

```c
#include <stdio.h>

void bubbla(int v[], int n) {
    int t;
    for (int i=0; i<n-1; i++) {
        printf("[%d]",v[i]);
        if (v[i]>v[i+1]) {
            t=v[i]; v[i]=v[i+1]; v[i+1]=t;
        }
    }
}

int main(void) {
int v[]={1,9,4,10,7,3,6,2,14,13,8};

bubbla(v,11);

for (int i=0; i<11; i++) printf("[%d]",v[i]);
}
```

Programming In Large…

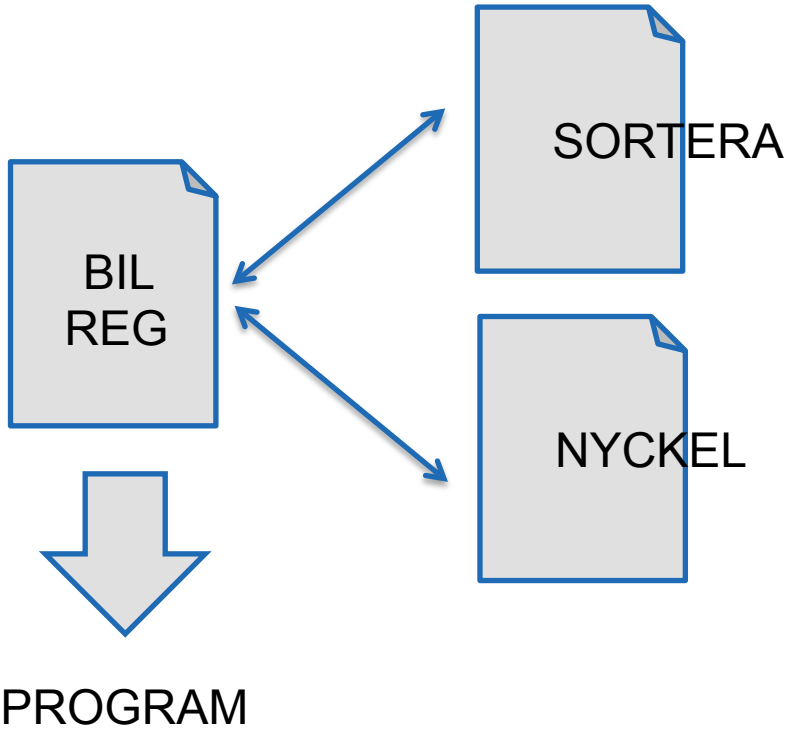
Module

Classes

#include < >

static

makefile



Både språket och programutvecklingsmetoden måste stödja tekniken!

Assembler-

Programspråk & programmering…

qualifiers

1
2
3

and
ior
xor
not
-
+

typedef

Sequencing

casting

Operands

Selection

1
2 eller 3
4

Branch
Skip

Reference

Repetition

Abstract

Aggregates

Goto

Jump

Compound

7
-2
14

Literals

DATA TYPES

CTRL STRUCT

Functions

Call
Return

Basic

Exceptions

Side effects

PRG CORR

PRG IN LARGE

class

main

Und. behavior

direktiv

#

Imp. dependent

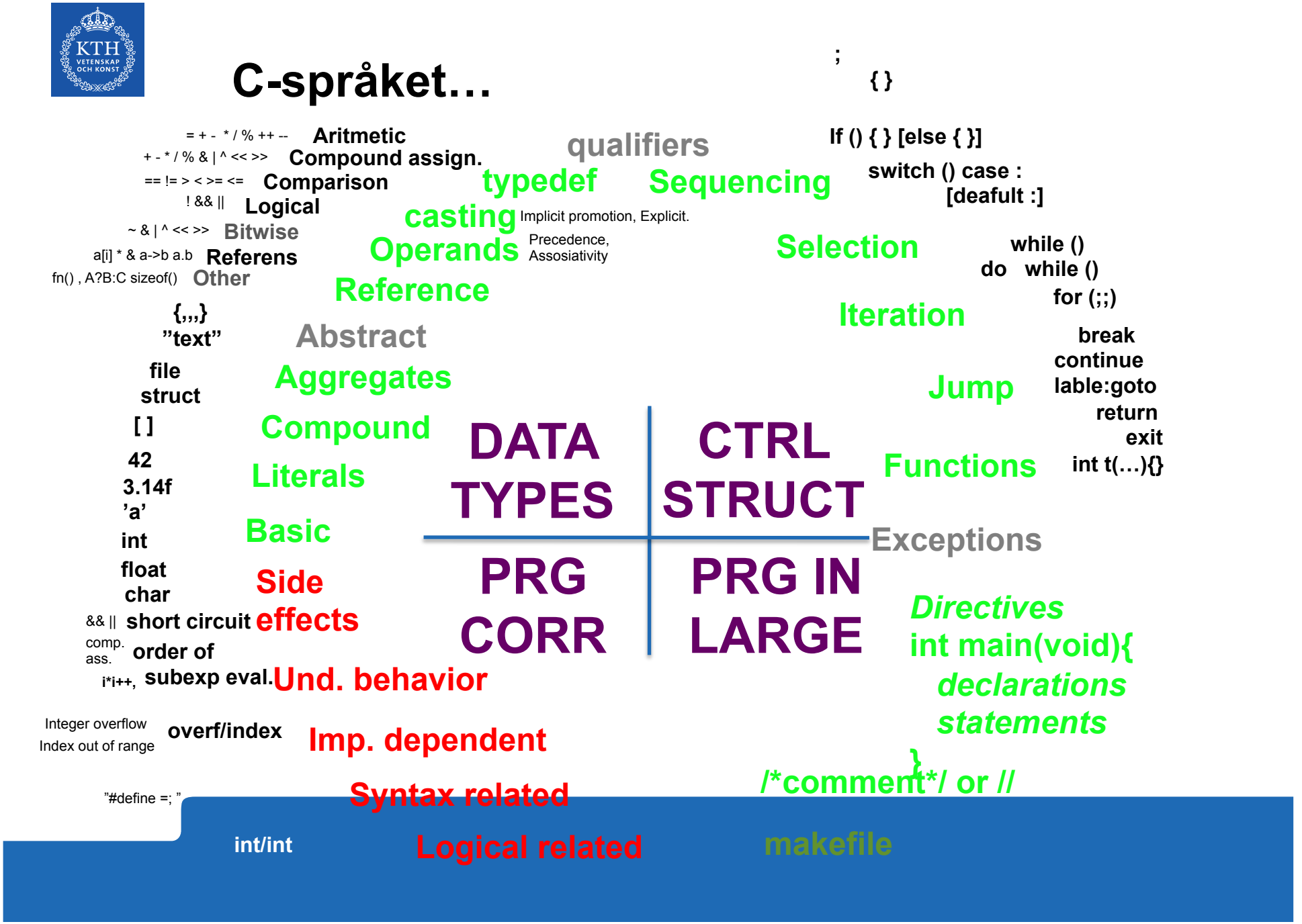
/*comment*/ or //

Syntax related

Logical related

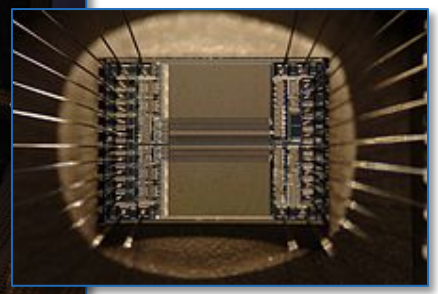
makefile

$A_1=1$
$A_2=1$
$A_n=A_{n-1}+A_{n-2}$


C-språket…
;
{ }
= + - * / % ++ --
Aritmetic
+ - * / % & | ^ << >>
Compound assign.
== != > < >= <=
Comparison
! && ||
Logical
~ & | ^ << >>
Bitwise
a[i] * & a->b a.b
Referens
fn() , A?B:C sizeof()
Other
qualifiers
typedef
Sequencing
If () { } [else { }]
switch () case :
[deafult :]
casting
Implicit promotion, Explicit.
Operands
Precedence,
Assosiativity
Selection
while ()
do while ()
for (;;)
Reference
Iteration
{,,,}
”text”
Abstract
break
continue
lable:goto
return
exit
int t(…){}
file
struct
Aggregates
Jump
[ ]
Compound
42
3.14f
’a’
Literals
DATA TYPES
CTRL STRUCT
Functions
int
float
char
Basic
Exceptions
Side effects
&& || short circuit
comp. ass. order of
i*i++, subexp eval.
PRG CORR
PRG IN LARGE
Directives
int main(void){
declarations
statements
}
Und. behavior
Integer overflow
Index out of range
overf/index
Imp. dependent
/*comment*/ or //
”#define =; ”
Syntax related
int/int
Logical related
makefile

Den tredje generationen: 1965-1974

IC revolutionen…



IBM360
-65—78
+1M op/s

Pipelining
Virtual mem.

Datorerna var nu så kraftfulla så flera använd. kunde dela på en dator. För detta krävdes ett kraftfullt styrprogram, de första OS växte fram!

Den tredje generationen: 1965-1974

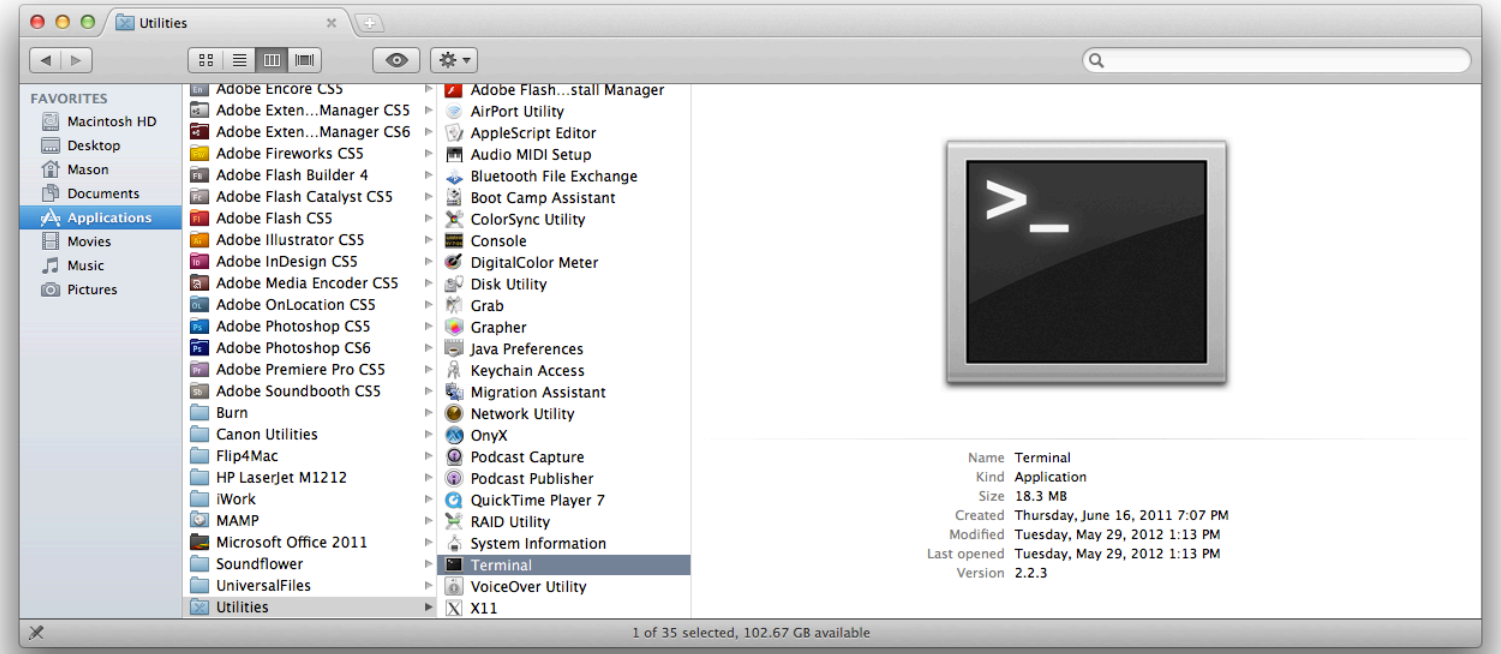
IC revolutionen…

- Operatören kommunicerade med OS:et via en kommandotolk!
- Den del i OS:et som hanterade dialogen benämndes vanligtvis Command Line Interpreter (CLI).
- Kommandona var ofta förkortningar.
- Varje datorleverantör med självaktning hade ett eget operativ-system med en egen CLI med egna, men likartade, kommandon.
- Speciellt viktigt var att kunna hantera filsystemet och vilka processer som kördes i datorn.

Trots att dagens datorer ofta har grafiska användargränssnitt, finns vanligtvis ett CLI kvar undangömt om det skulle behövas!

Den tredje generationen: 1965-1974

IC revolutionen…



Windows: Command Prompt

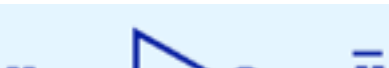
Den tredje generationen: 1965-1974

IC revolutionen…

	Mac/Linux	Windows (DOS)
Hjälp	**man**	**help**
Visa filerna som finns där jag är	**ls**	**dir**
Där jag är i filkatalogen	./	.\
En nivå upp i filträdet	..	..
Flytta en nivå upp i filträdet	**cd** ..	**cd** ..
Hemkatalogen	**cd** ~/	
Flytta till specifik plats	**cd** path	**cd** path
Flytta en fil	**mv** from to	**move** from to
Kopiera en fil	**cp** from to	**copy** from to
Radera en fil	**rm** path	**del** path
Visa innehållet i en fil, sida för sida	**more** path	**more** path
Styr om output, input, länka (pipe)	> < |	> < |

Den tredje generationen: 1965-1974

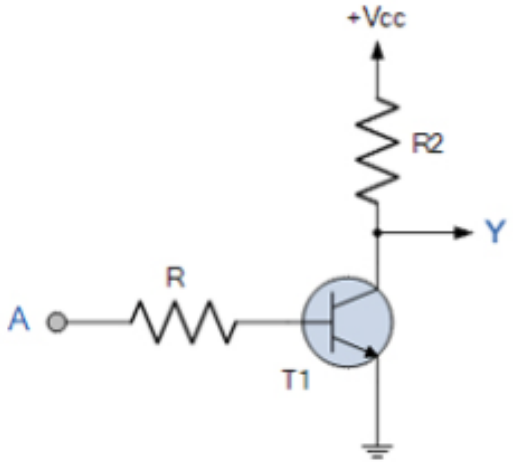
IC revolutionen…

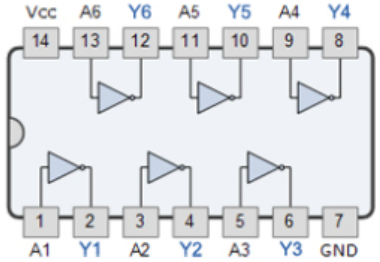


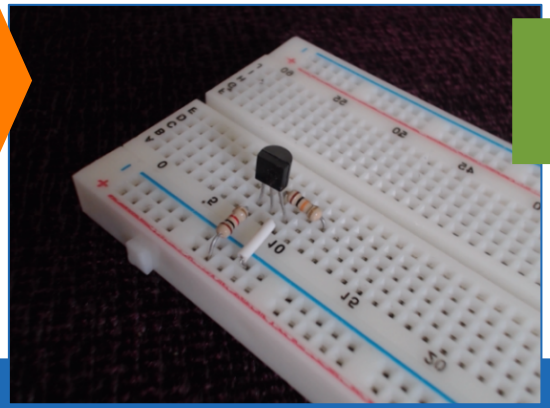
NOT x

x	$\overline{x}$
0	1
1	0

IBM NOT Gate







Den tredje generationen: 1965-1974

IC revolutionen…

SPECIAL
APOLLO 11

55W

2 k RAM
36k ROM
2 MHz

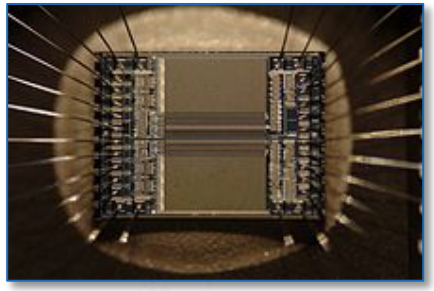
```
1088   REF    1                                          COUNT*  $$/RS2
1089                          14,3671    77420  1  RS2   STQ     EXIT
1090   REF    1               14,3672    03665  1        SAVERS2
1091   REF   10  LAST   946   14,3673   51*757  0        INDEX   STARIND
1092   REF    9  LAST   935   14,3674    3 1755  1        CA      BESTI       PICK UP STARCODE DETERMINED BY RS6
1093                          14,3675    0 0006  1        EXTEND
1094   REF    1               14,3676    3 3772  1        MP      1/6TH**
1095   REF   33  LAST   900   14,3677    6 4744  1        AD      BITS        SET DETENT POSITION 2
1096   REF    2  LAST   100   14,3700   54 734   0        TS      STARCODE    SCALE AND STORE IN STARCODE
```

4nW

Margaret Hamilton
Chefsprogrammerare

Den tredje generationen: 1965-1974

IC revolutionen…



Miniräknare!

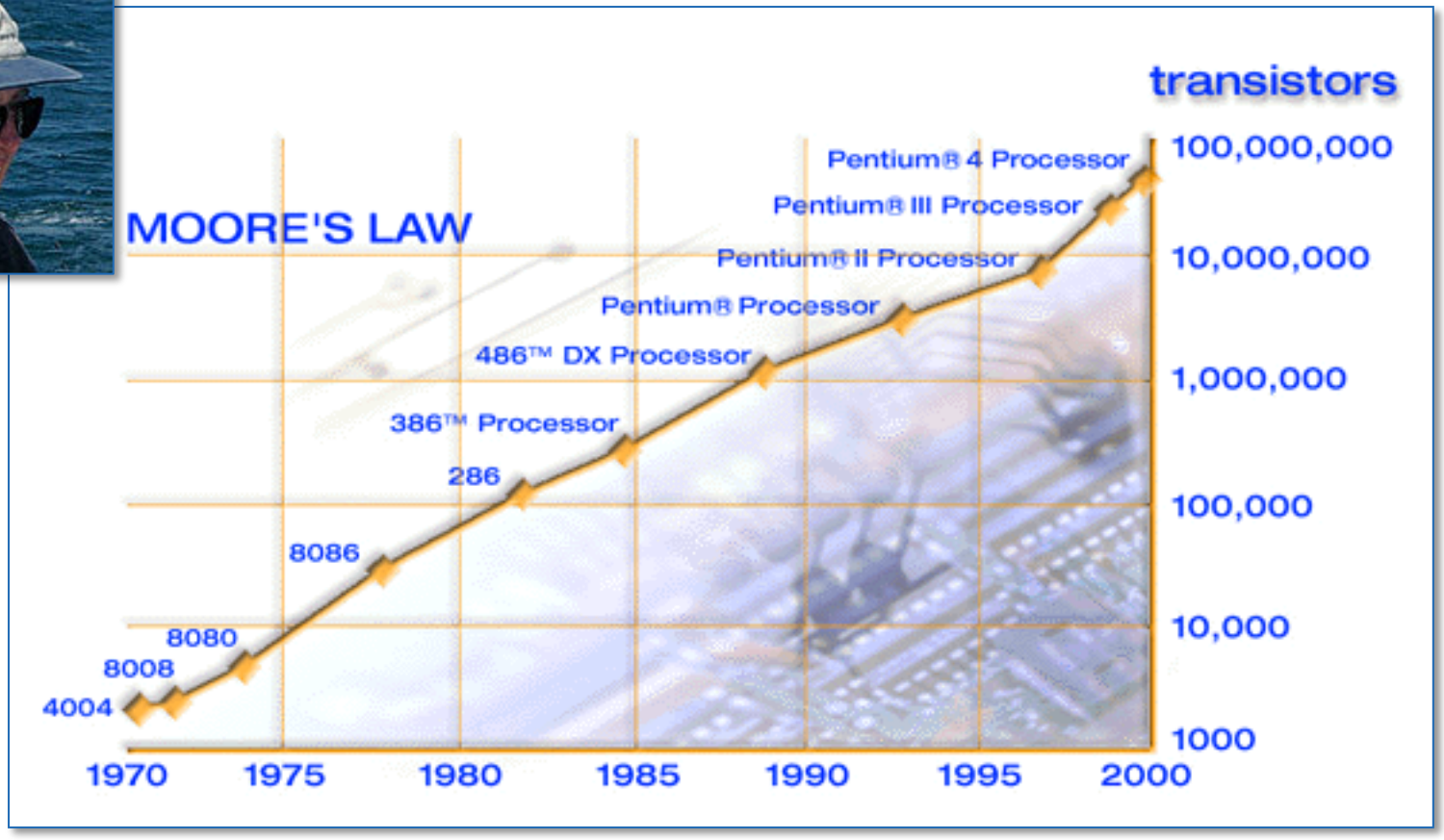
-71 4004
-73 8080

Mot slutet av perioden gick det att få in så många transistorer på en IC så att funktionen till en hel enkel dator kunde skapas…

Den fjärde generationen: 1975-1983

Mikrodatorrevolutionen…

Gordon -04

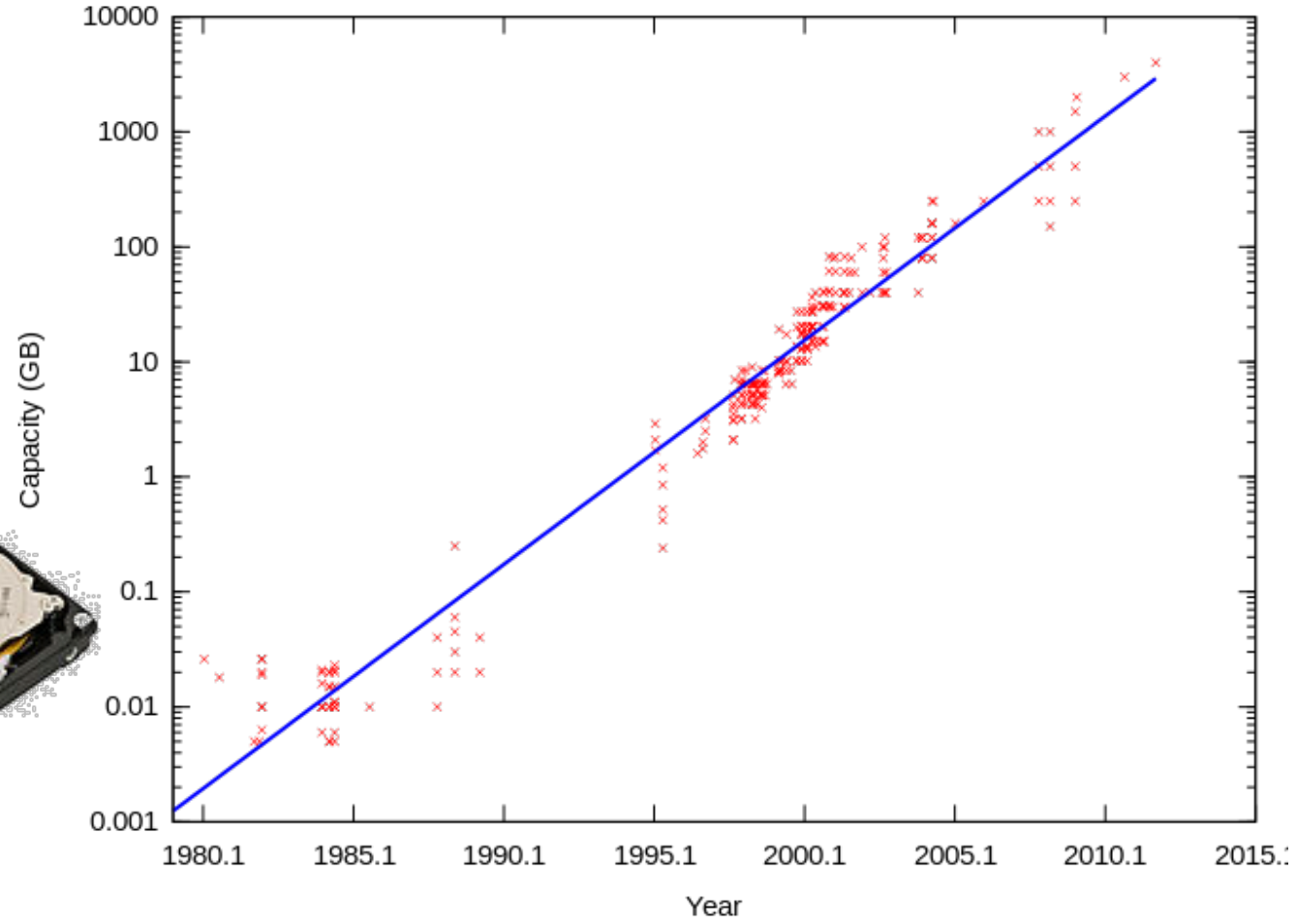


Stordatorerna började ersättas med datorkraft på skrivbordet + centralt filsystem

Timesharing VT100

Den fjärde generationen: 1975-1983

Mikrodatorrevolutionen…



HD, FD…

Undantag: Batterier!

Den femte generationen: 1984-1992

Persondatorrevolutionen…

-82: Commondore 64

-83: IBM PC 5150

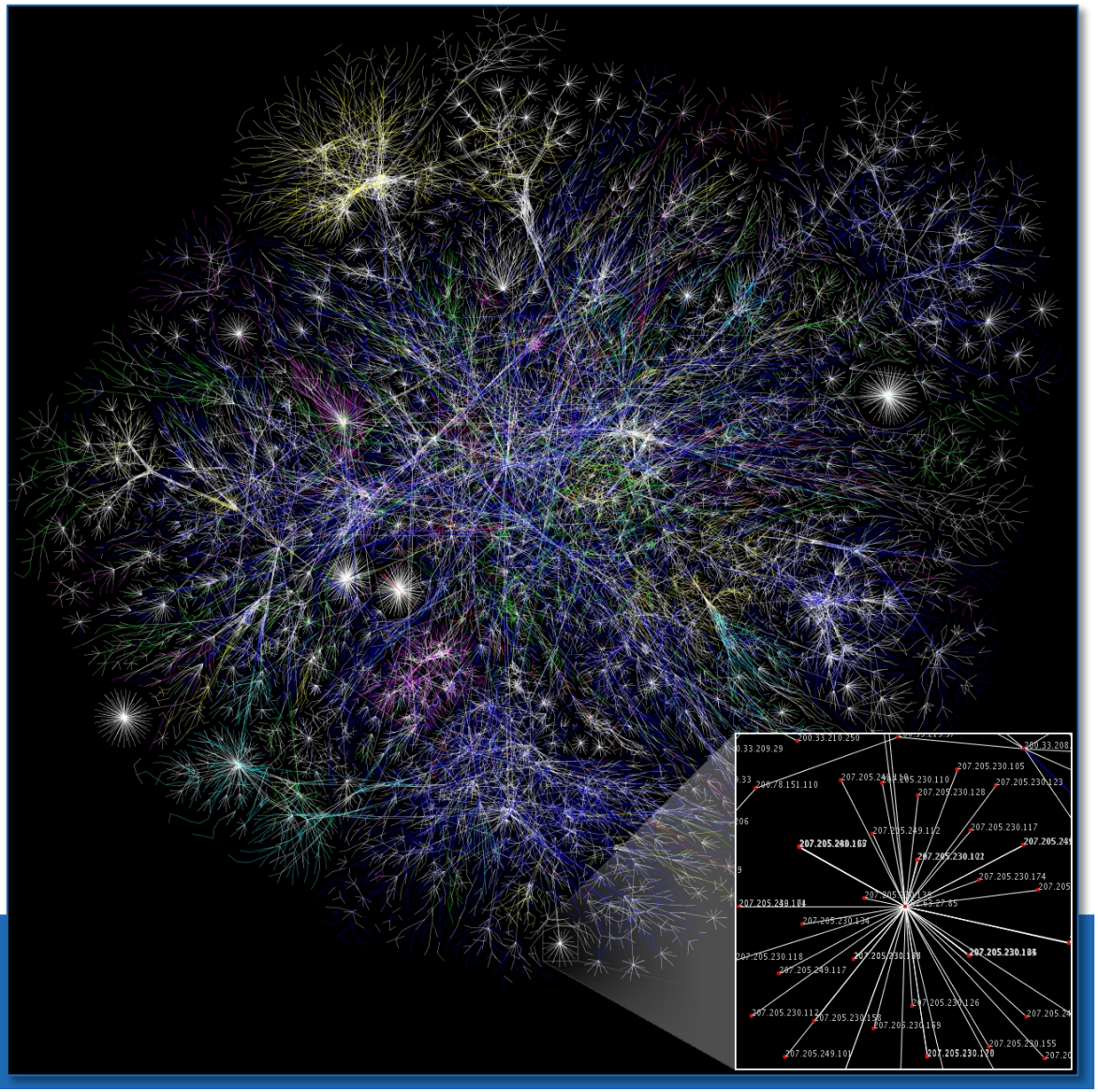
-84: Apple Machintosh

Grafiskt användargränssnitt (GUI)

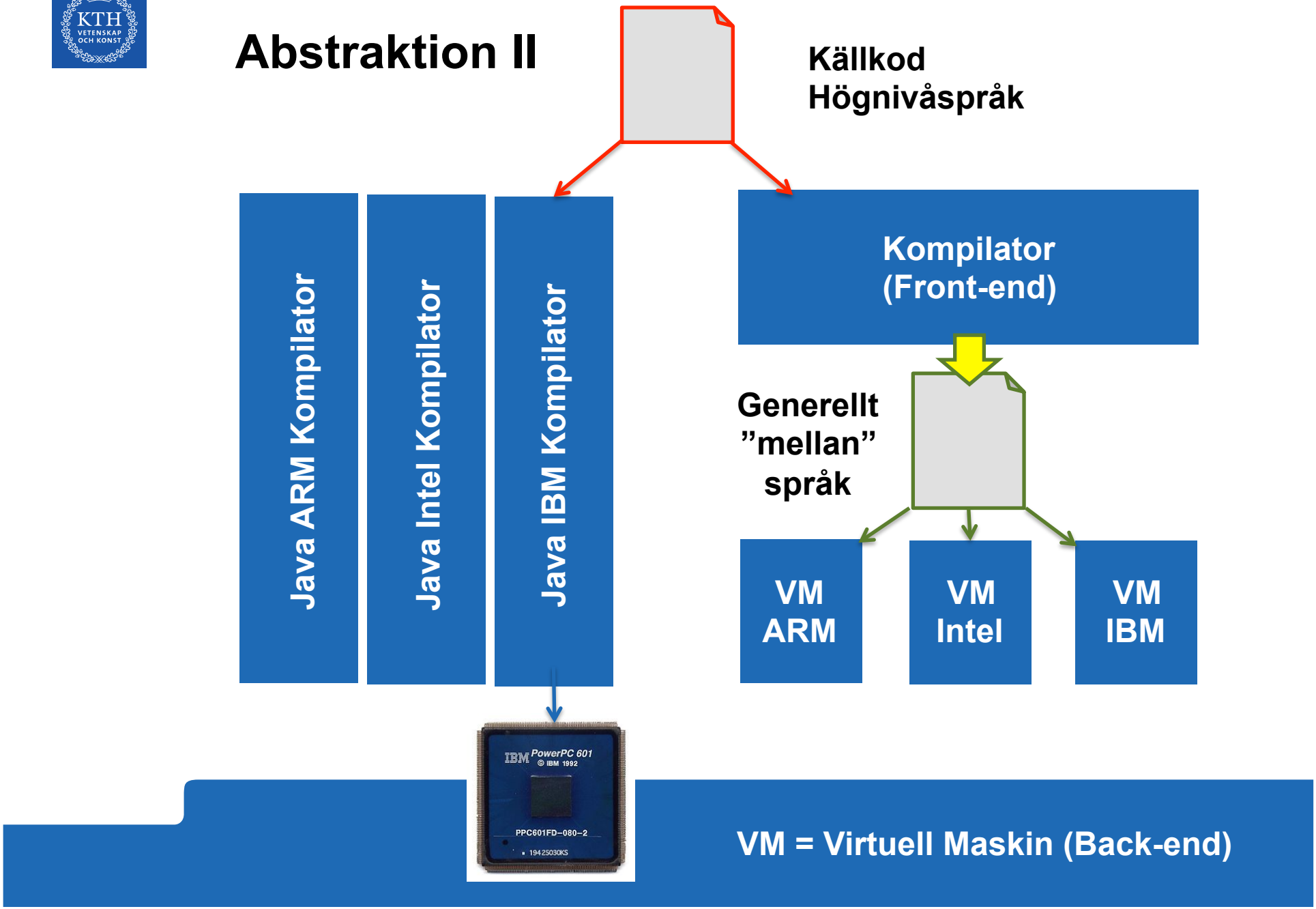
Den sjätte generationen: 1993-2004

Internetrevolutionen…

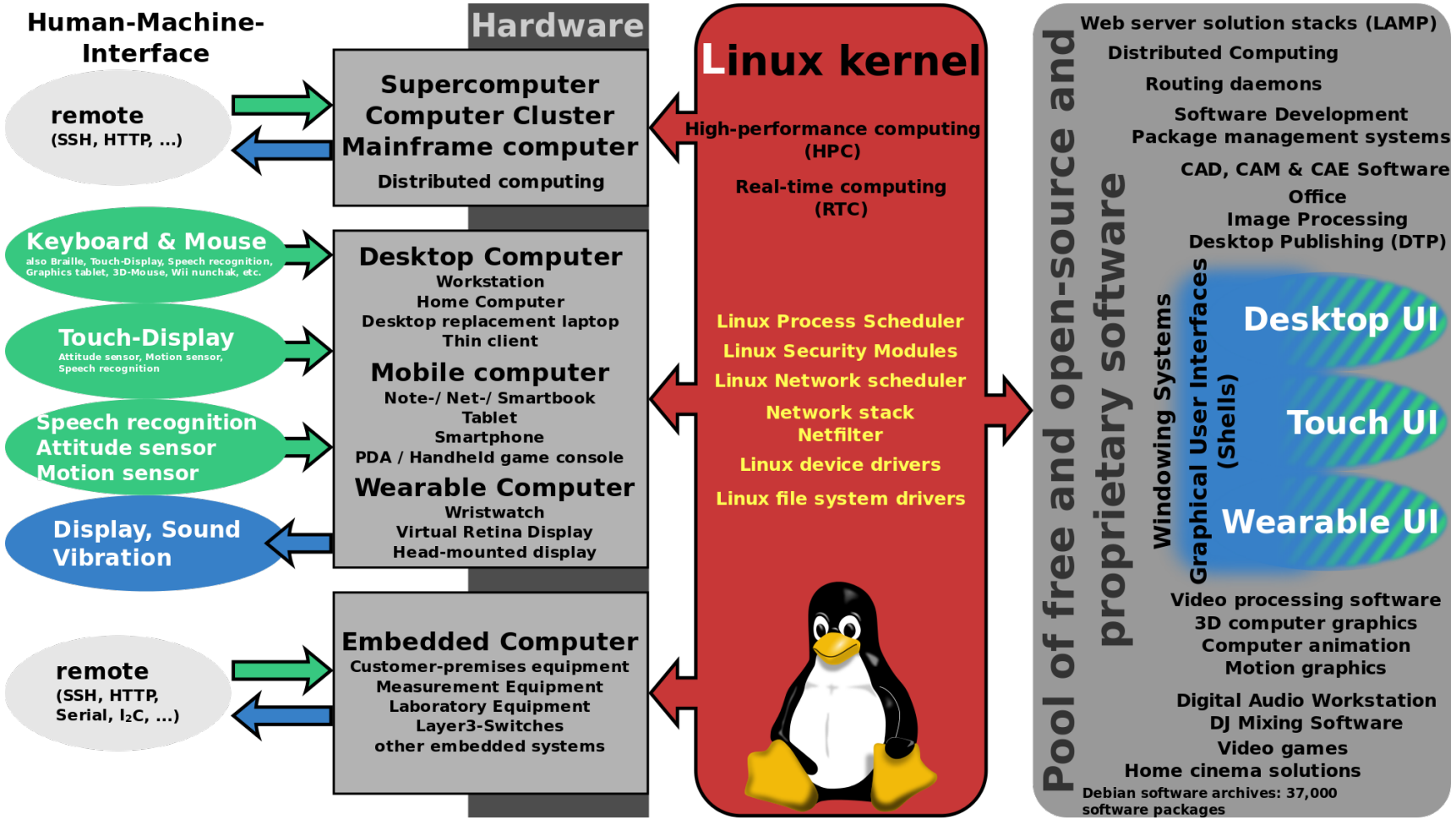
Sir Timothy John "Tim" Berners-Lee
2014



Abstraktion II



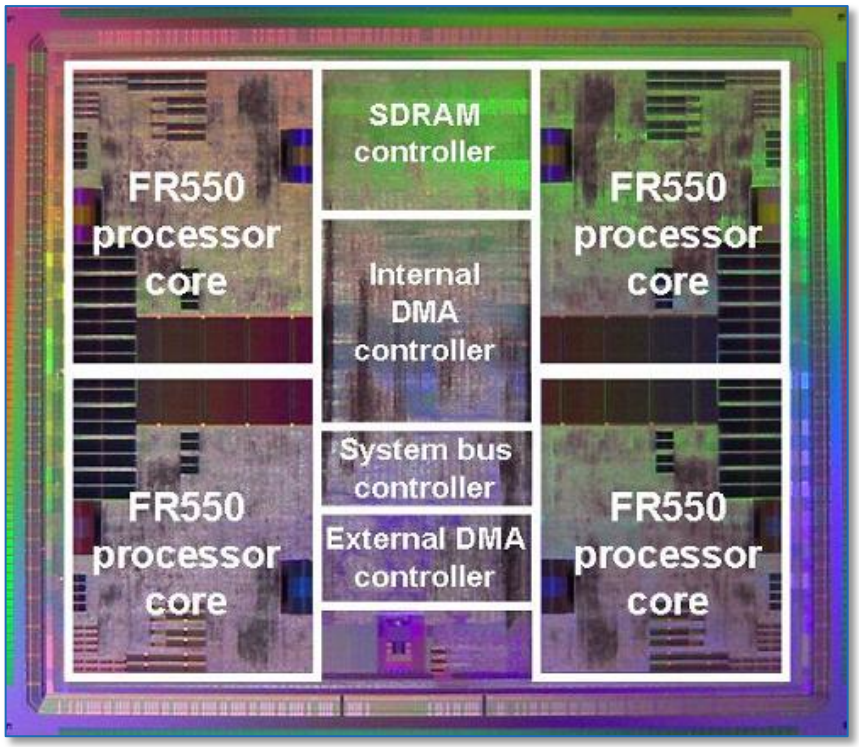
Abstraktion III



Ca 4000-rader kod måste uppdateras för att flytta Linux till en ny plattform! Ett ”helt” operativsystem är miljontals rader kod.

Den sjunde generationen: 2005-2015

Streaming/flerkärnerevolutionen…



Fujitsu -05 (4/100k)

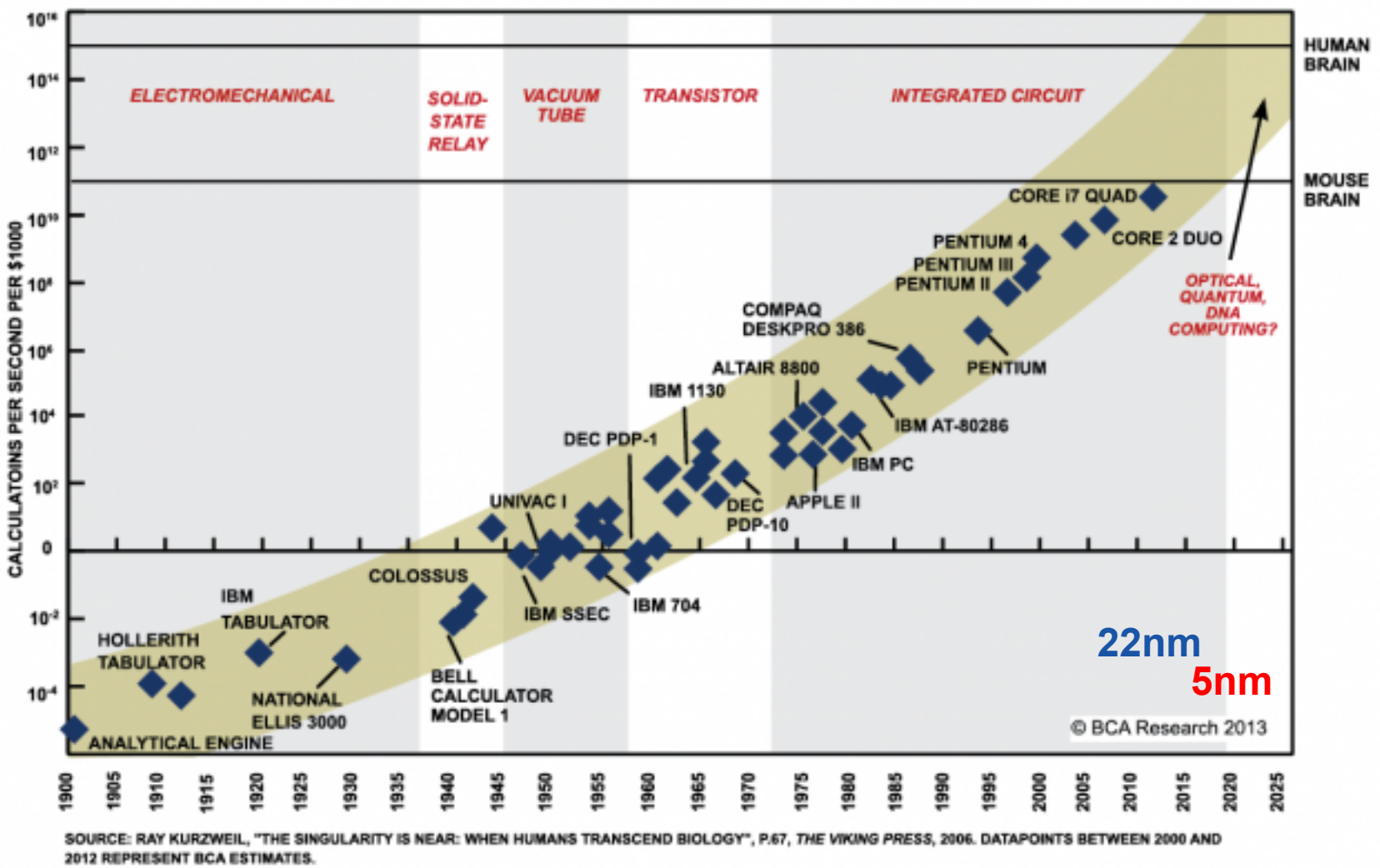
YouTube

Spotify®

Programmeringsspråk med stöd för parallella processer…

Den åttonde generationen: 2016-

Den biologiska/nano revolutionen…



SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

Vi ska lära oss programmera…

…varför tycker en del att det är svårt?

Ett programmeringsspråk består vanligtvis av ca 50 ord…

...det kan omöjligtvis vara svårt att lära sig 50 engelska ord...

...men det är knepigt att beskriva verkligheten med bara 50 ord...

...tricket är att beskriva verkligheten i små små steg...

...och att inte glömma bort att datorn fungerar...

...precis som du, när du läser ett recept!

Hur använder man (dator-)språket?

Arbetsintensiv *Monoton*

Konsten att översätta en **verklig process** till datorns värld…

Farlig *Felkänslig*

Väg 4 tomater och beräkna medelvärdet!

Hur använder man (dator-)språket?

Arbetsintensiv *Monoton*

Konsten att översätta en **verklig process** till datorns värld…

Farlig *Felkänslig*

Väg 4 tomater och beräkna medelvärdet!

Nollställ miniräknaren.

Repetera 4 gånger:
- **Väg nästa tomat…**
- **…addera vikten till summan på miniräknaren.**

Dividera summan med 4!

Hur använder man (dator-)språket?

Arbetsintensiv *Monoton*

Konsten att översätta en **verklig process** till datorns värld…

Farlig *Felkänslig*

Väg 4 tomater och beräkna medelvärdet!

Nollställ miniräknaren.

Repetera 4 gånger:
Väg nästa tomat…
…addera vikten till summan på miniräknaren.

Dividera summan med 4!

```
int measurement;
int total = 0;

for (int counter=0; counter<4; counter++) {
    scanf("%d",&measurement);
    total+=measurement;
}

printf("Avrage = %f \n", total/4.0);
```

…det är att kunna programmera, och rätt lika från språk till språk!

Prova på att beskriva ”gissa talet”!

Prova på att beskriva ”gissa talet”!

Generera ett slumptal!

Välkommen till spelet!

Gissa ett tal?
Läs in talet!

Om =, skriv ut **Rätt!**
Om >, skriv ut **För högt!**
Om <, skriv ut **För lågt!**

Fortsätt om ej =

Prova på att beskriva ”gissa talet”!

Generera ett slumptal!

Välkommen till spelet!

Gissa ett tal?
Läs in talet!

Om =, skriv ut **Rätt!**
Om >, skriv ut **För högt!**
Om <, skriv ut **För lågt!**

Fortsätt om ej =

```c
/* Iterators 0.4 AC */
#include <stdio.h>
#define X 6

int main(void){
    int n;          /* Guess */

    printf("Iteration Tester (IT)!\n");

    do {
        printf("Guess a number :");
        scanf("%d",&n);

        if (n==X) {
            printf("! CORRECT !\n");
        } else {
            if (n>X)
                printf("! To High !\n");
            else
                printf("! To Low  !\n");
        }
    } while (n!=X);

    printf("IT Done.\n");
    return 0;
}
```

Prova på att beskriva ”gissa talet”!

Generera ett slumptal!

Välkommen till spelet!

Gissa ett tal?
Läs in talet!

Om =, skriv ut **Rätt!**
Om >, skriv ut **För högt!**
Om <, skriv ut **För lågt!**

Fortsätt om ej =

```c
/* Iterators 0.4 AC */
#include <stdio.h>
#define X 6

int main(void){
    int n;          /* Guess */

    printf("Iteration Tester (IT)!\n");

    do {
        printf("Guess a number :");
        scanf("%d",&n);

        if (n==X) {
            printf("! CORRECT !\n");
        } else {
            if (n>X)
                printf("! To High !\n");
            else
                printf("! To Low  !\n");
        }
    } while (n!=X);

    printf("IT Done.\n");
    return 0;
}
```

```
Iteration Tester (IT)!
Guess a number :5
! To Low  !
Guess a number :7
! To High !
Guess a number :6
! CORRECT !
IT Done.
```

Förstår du det här blir resan lätt!

Föreläsning #3

HI1026 Proj
HI1007 Java
HI1024 C
HI1030 DB
HI1029 DS
HI1031 Dist
HE1034 Nät
HE1033 Nät
HI1032 Nät
HE1037 Nät
HI1025 OS


GUI

CLI

Program #1 | Program #2 | * | .html / Browser | IoT Prg

t.ex Java

KOMPILATOR

t.ex bytekode

OPERATIV **OS** SYSTEM

Virtuell Maskin

Skärm tgb **Standard Bibliotek** *Filsys. Nätverk*

Processer Minne **Kärna** *Resurshantering*

RT OS ---- IoT Prg

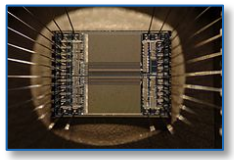
Assemblerkod

ASSEMBLER

Maskinkod

HE1028 uC
HE1041 uC

Maskinkoden utgör ett abstrakt gränssnitt som döljer bort hur den underliggande hårdvaran är uppbyggd…



HE1026 Dig

…av ett antal IC-kretsar med miljontals grindar…

HE1030 Ana

…som i sin tur består av miljardtals transistorer!
(som trots allt inte kan göra mycket mer än 1+1!)

Lab #1: Hour of code – Minecraft!

Minecraft-kodtimmen

Använd block av kod för att ta Steve eller Alex på ett äventyr genom denna Minecraft-värld.

Lärarens anteckningar
https://hourofcode.co...

Kör

code.org

Lab #2:Lite svårare program…

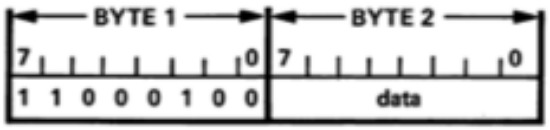
Klassisk Labyrint 20 övningar!

Rev history

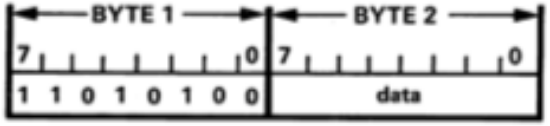
150301 0.1 AC	Skapad.
150507 0.8 AC	Utkast #1, remiss NB.
150813 0.9 AC	Bättre introduktion.
150818 1.0 AC	Bättre avslutning + övning.
160614 1.1 AC	Inkluderat modern datorhistoria från föreläsning #1. Flyttat programutvecklingsmetodik till föreläsning #3. Fördjupat begreppen abstraktion och operativsystem.
160624 2.0 AC	Uppsnyggat inför HT16
170725 3.0 AC	Adderat CLI avsnitt

IMMEDIATE INSTRUCTIONS

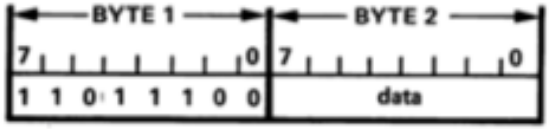
LOAD IMMEDIATE LDI(C4)



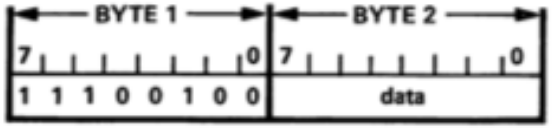
AND IMMEDIATE ANI(D4)



OR IMMEDIATE ORI(DC)

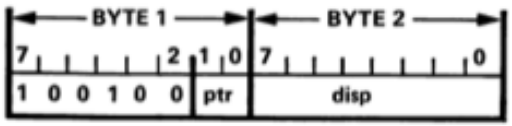


EXCLUSIVE-OR IMMEDIATE XRI(E4)

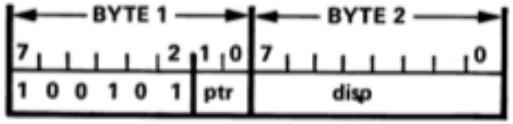


TRANSFER INSTRUCTIONS

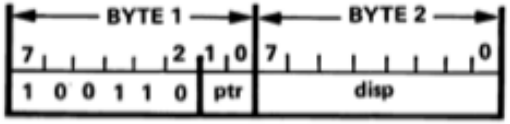
JUMP JMP(90)



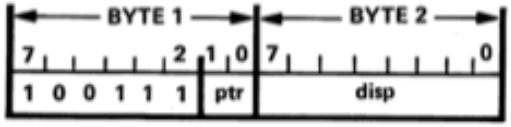
JUMP IF POSITIVE JP(94)



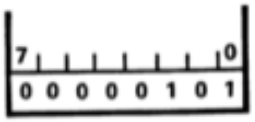
JUMP IF ZERO JZ(98)



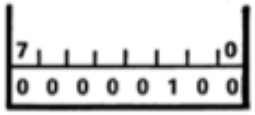
JUMP IF NOT ZERO JNZ(9C)



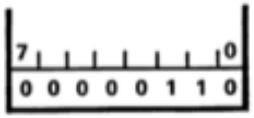
ENABLE INTERRUPT IEN(05)



DISABLE INTERRUPTS DINT(04)



COPY STATUS TO AC CSA(06)



COPY AC TO STATUS CAS(07)