

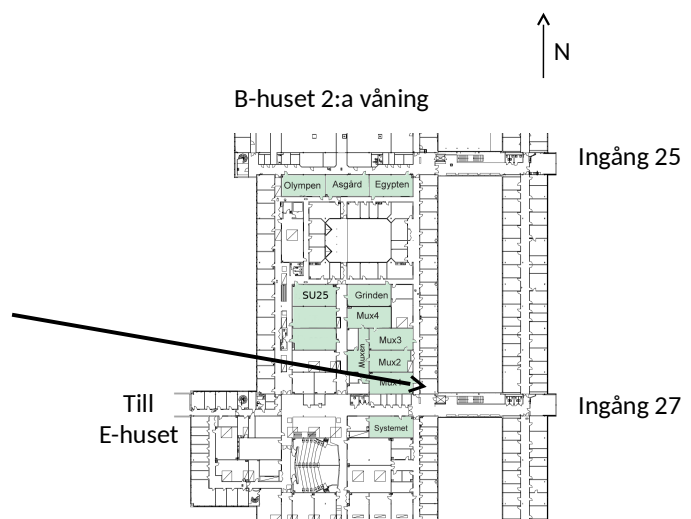
TSEA28 Datorteknik Y (och U)

Föreläsning 1

Kent Palmkvist, ISY

Vem är jag

- Föreläsare och kursansvarig
 - Kent Palmkvist
 - Kent.Palmkvist@liu.se
 - Kontor 3B:502 (andra våningen)



Tillgänglig information

- Alla slides finns på kursens hemsida
 - <https://www.isy.liu.se/edu/kurs/TSEA28>
- Kursmaterial även på kurshemsidan
 - Anvisningar för kursen (deadlines etc.)
 - Labbmateriäl
 - Gamla tentor
 - Material från tidigare versioner av kursen
- Lisam används också
 - Labanmälan

Kursens mål

- | | |
|---|---|
| <ul style="list-style-type: none">• Förstå<ul style="list-style-type: none">- Hur en dator är uppbyggd och fungerar på registernivå- Binär aritmetik• Kunna<ul style="list-style-type: none">- Skriva små enkla assemblerprogram- Hantera in/utmatning- Implementera instruktioner m h a mikroprogrammering | <ul style="list-style-type: none">• Känna till<ul style="list-style-type: none">- Principer för hur cache fungerar- Olika sätt att öka exekveringshastigheten- Hur processorn påverkar operativsystemet |
|---|---|

Varför detta är intressant

- En programmeringsintresserad U:are
 - Förstå begränsningar hos olika datorer
- En matematikintresserad Y:are
 - Snabba upp beräkningsprogram och simuleringar
- En fysikintresserad Y:are
 - Använda sensorer för insamling av mätdata
- En elektronikintresserad student
 - Nästan all elektronik innehåller en dator

Det är roligt!

Senare kurser

- För Y
 - Elektronik kandidatprojekt
 - Datorteknik och realtidssystem
 - Inbyggda DSP processorer
 - Datorarkitektur
- För U
 - Processprogrammering och operativsystem
 - Datorarkitektur
 - Kompilatorkonstruktion

Administrativ information

Kursens innehåll

- Går VT1 + VT2
- 6 hp (3 hp tentamen, 3 hp lab)
- 16 föreläsning (8 under VT1)
- 4 lektioner (2 under VT1)
- 5 laborationer (3 under VT1)
- 1 digital tentamen (4h)
- Självstudier > 100h !!
 - Laborationsförberedelser ~ 12h per lab
 - Varierar mellan studenter, vissa kan behöva mer tid!

Kurslitteratur

- Computer Organization and Design, ARM Edition, D. A. Patterson, J.L. Hennessy, ISBN 9780128017333, ebook 9780128018354
 - Referenslitteratur
 - Ny för i år
- Föreläsningsslides täcker samma innehåll
 - Lägg upp på lisam och websidorna
- Går att genomföra kursen utan att skaffa boken



Kurslitteratur, gamla boken

- Computer Organization and Architecture – Themes and Variations; Alan Clements
 - Trycks inte längre
- Många alternativ finns
- Föreläsningsslides täcker samma innehåll
 - Lägg upp på lisam och websidorna



Kurslitteratur, övrigt material

- Laborationsanvisningar (finns på kurshemsidan)
 - Referensmaterial för laborationshårdvara och i tentauppgifter
- Notering: Tidigare års (Roos mfl) kurslitteratur går att använda, men saknar många detaljer
 - Grundläggande Datorteknik (Roos)
 - Kompendium i Datorteknik (Wiklund)

Föreläsningar och lektioner

- Föreläsningar bygger delvis på kursboken, men även annat material ingår
 - Slideskopior finns på kurshemsidan
 - Går att klara sig utan kursbok
- Lektioner introducerar och beskriver labbuppgifterna
 - Se dom som frågestunder
 - Se till att vara förberedd innan lektion!
 - Läs igenom labb-beskrivning och dokumentation innan lektion!

Laborationer

- Fem laborationer uppdelat på 2 x 2h (totalt 4h per labb)
 - Lab 1: Kodlås (introduktion till assembler)
 - Lab 2: Avbrott
 - Lab 3: Digitalur
 - Lab 4: Mikroprogrammering
 - Lab 5: Bussar och cache
- Laborationsgrupper
 - Max 2 personer per grupp
 - Max 15 grupper per labbtillfälle
- Labbanmälan via lisam
 - Måndag 27/1 kl 12.45 öppnar anmälan
 - Alla tider kanske inte passar alla!
- Viktigt!
 - Labbanmälan stänger dagen innan första labbtillfället
 - Bara anmälda studenter får delta på laborationen

Laborationsförberedelser

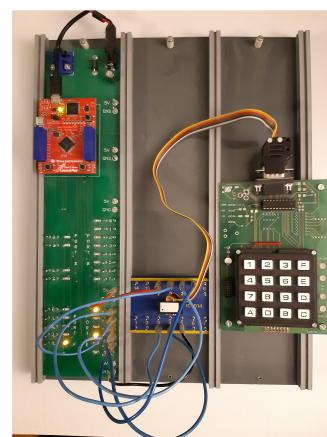
- Förbered innan labbtillfälle och lektion
 - Läs labkompendie och annat material
 - Lös uppgifter och/eller skriv kod innan labbtillfället
- Använd gärna möjligheten att köra utrustningen hemifrån
 - Möjligt köra distansversionen för att se om kod fungerar
 - Tillgängligt redan nu, se anvisningar på kurshemsidan
- Redovisning av lab görs på plats
 - Närvaro krävs vid redovisningen (går inte sitta på distans)
 - Inkluderar uppkoppling och test
- Lab på plats tillgängligt utanför schemalagd tid, start efter 1:a labtillfället

Förändringar från tidigare år

- Ny version av examinationsform: Digital Tentamen med Inspira
 - Skriv svar på dator (inspera)
 - Endast dator som texteditor (inget stöd för kompilering/syntax highlight etc.)
 - Kräver windows eller mac dator (ej linux!)
- Nya bedömningskriterier
 - Finns att läsa på kursens lisam-sida
 - Lite ändrad struktur och krav på tenta
 - Minimikrav på poäng på vissa frågor

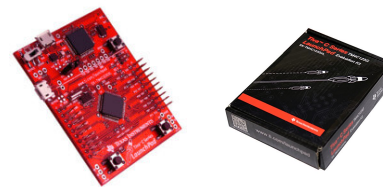
Laborationshårdvara Lab 1-3: Darma

- System med mikrokontroller baserad på Cortex-M4 processor
 - Tillverkare: Texas Instruments (det lilla röda kortet)
 - Processordesign: ARM
 - Gröna kortet till vänster: Darma, lokalt LiU
 - Knappar, displayer etc. kopplas in via kopplingspunkter i det gröna kortet
- Mjukvaran för programmering gratis nedladdningsbart från ti.com
 - Fungerar på windows, mac och linux
 - För att testa kod behövs ett inkopplat kort



Labutrustning för arbete på distans

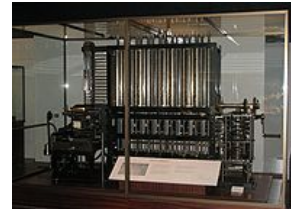
- Emulerar anslutna tryckknappar etc mha programvara
 - Fast koppling
- Styr och mät av värden på anslutningar via en Arduino Uno
 - GUI för att se och styra (skrivet i python)
- Design tillgänglig på gitlab.liu.se för dom som vill bygga eget eller bara är nyfikna
 - Röda kortet är ett EK-TM4C123GXL (200-300 kr)



Lite datorhistoria

De tidiga experimenten

- Analytic engine (1837, Charles Babbage)
 - Ritningar men ingen fungerande maskin
 - Ca 3 minuter för multiplikation av två 20-siffriga tal
- Z3 (1941, Konrad Zuse)
 - Elektromekanisk (relä)
 - 3 sekunder för multiplikation



https://commons.wikimedia.org/wiki/File:Babbage_Difference_Engine.jpg



https://commons.wikimedia.org/wiki/File:Z3_Deutsches_Museum.JPG

Transistorbaserad elektronik

- Eniac, 1946
 - 18 000 radiorör,
150 kW effektförbrukning
 - 1 kW ~ en värdefläkt hemma
 - 5000 additioner/subtraktioner per sekund
- Intel 4004, 1971
 - 2300 transistorer
 - Grundläggande komponent ~1 elektrisk switch
 - Första enchips mikroprocessorn
 - 740 kHz klockfrekvens, 0.095 MIPS (Miljoner Instruktioner Per Sekund)



https://commons.wikimedia.org/wiki/File:ENIAC_Penn1.jpg#/media/File:ENIAC_Penn1.jpg



https://commons.wikimedia.org/wiki/File:Legendary_Chip_Designer_Betting_on_Human_Mind.jpg#/media/File:Legendary_Chip_Designer_Betting_on_Human_Mind.jpg

Datorer 1970-tal

- Cray-I, 1975
 - Superdator, vektormaskin, 80 MHz, 160 MIPS
 - Väderprognoser, designarbete flygplan etc.
- Hemdatorer börjar dyka upp
 - Apple II, 1977, resp. VIC20, 1981 (tidiga hemdatorer)
 - 8-bitars 6502 processor
 - 1 MHz klockfrekvens, 3500 transistorer
 - 48 Kbyte RAM minne (max)



<https://commons.wikimedia.org/wiki/File:Commodore-VIC-20-FL.jpg#/media/File:Commodore-VIC-20-FL.jpg>

Två viktiga designer som påverkar än

- IBM PC, 1981
 - 16-bitars 8088 processor, 29000 transistorer
 - 4.77 MHz klockfrekvens, max 640 KB RAM
 - Ursprunget till PC-datorer (windows-maskiner)
- ARM, 1985
 - 32-bitars RISC-processor, 25000 transistorer
 - Acorn Archimedes
 - ARM Ltd startat av Apple och Acorn 1990



https://commons.wikimedia.org/wiki/File:IBM_PC-IMG_7271.jpg#/media/File:IBM_PC-IMG_7271.jpg



<http://www.oldcomputers.net/ibm5150.html>



<https://commons.wikimedia.org/wiki/File:AcornArchimedes-Wiki.jpg#/media/File:AcornArchimedes-Wiki.jpg>

Början av 2000-talet, större effektutveckling

- AMD64, 2003
 - 64-bitars 8086 kompatibel
 - 100 miljoner transistorer
 - 1 GHz klockfrekvens
- Intel Haswell (i3, i5, i7), 2013
 - Inkluderar grafikprocessor, 1.4 miljarder transistorer, 3 GHz
- AMD Epyq, 2017
 - 32 kärnor (processorer), 19 miljarder transistorer, 180W



En AMD-processor utan kylare brinner upp...

Effektutveckling blir en viktig begränsning för persondatorer runt år 2000

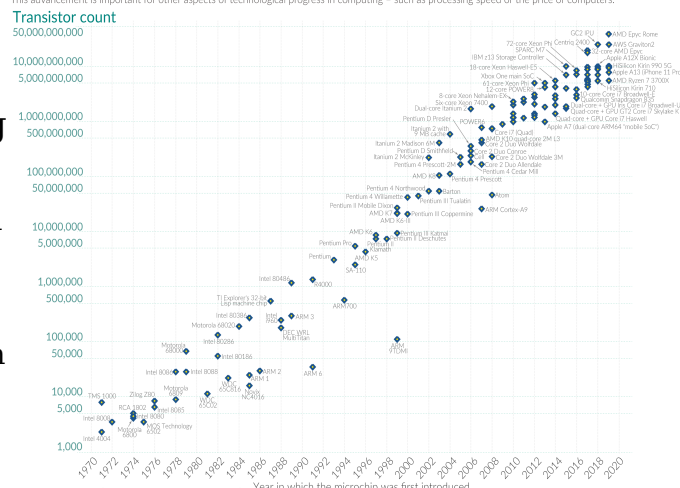
Bärbara/mobiler

- Mobila enheter har också en ökande komplexitet
 - Inkluderar oftast grafikgenerering, 5G och WLAN
- Numera även stöd för avancerade kameror och AI-applikationer
- Apple A5 (Iphone 4S), 2011
 - 2 kärnor, 800 Mhz
- Apple A11 Bionic (Iphone 8), 2017
 - 6 kärnor (2 stycken på 2.4GHz), 4.3 miljarder transistorer
- Apple A12 Bionic (Iphone XR), 2018
 - 6 kärnor (2 stycken på 2.5GHz), 7 miljarder transistorer

Lite datorhistoria, trender

- Miniaturisering ger allt snabbare datorer
- Exponentiell prestandaökning
 - Moores lag
 - Bildens y-axel visar exponentiell skala (antal transistorer)
- Fysiska begränsningar finns
 - Tillverkningsprocessernas namn speglar inte direkt de fysiska måtten (7 nm, 5 nm etc.)

Moore's Law: The number of transistors on microchips doubles every two years
 Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
 OurWorldinData.org – Research and data to make progress against the world's largest problems. Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.

Prestandamått. Aktuella trender

- Beräkningshastighet
 - Instruktioner / s (MIPS)
 - Operationer / s (TOPS)
- Effektförbrukning (joule/instruktion)
 - Värmeutveckling ger krav på kylning
- Större chip, mindre transistorer
 - Fler transistorer och längre ledare ger större kapacitanser och större effektförbrukning
- Hårdvaran allt mer specialiserad
 - Kryptering, videokodning och avkodning, Signalbehandling
- Många parallellkopplade datorer
 - Ibland delar datorerna vissa delar, (inkl delar av processorn)
- Mjukvaran måste anpassas till denna hårdvara
 - Kräver god förståelse för datorteknik
- Kostnad/transistor minskar

Summering

- Hastigheten (instruktioner per sekund) ökar exponentiellt
- Priset går ned trots ökad prestanda
- Moores lag
 - Antal transistorer på ett chip dubblas vartannat år (börjar gå långsammare)
- Generellt: Prestanda ökar exponentiellt
 - Olika exponenter => differensen ökar också exponentiellt
 - Exempel: Under 1980-talet tog det lika lång tid för processorn att utföra en instruktion som en läsning/skrivning i minnet. Antal instruktioner/sekund har ökat fortare än antal läsningar/sekund i minnet

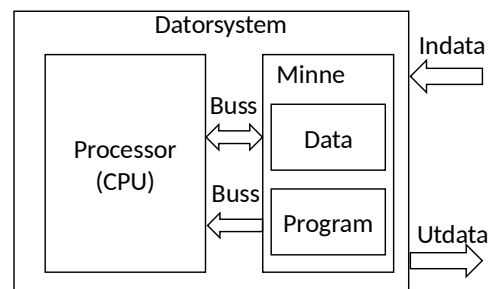
En dators inre uppbyggnad

En programmerares vy av en dator

- Högnivåspråk
 - Interpreterande (tolka kod under körning, t ex java, python)
 - Kompilerande (översätt innan körning till maskinkod, t ex C)
 - Generella, går att använda på många olika datorer
- Assemblerspråk (maskinkod)
 - Datorns egna interna språk, unikt för varje datorfamilj
 - Alla högnivåspråk måste översättas till eller tolkas av program beskrivna i assembler
- Mikroprogrammering (inget en vanlig användare ser)
 - Styrning av interna funktioner i datorns inre

Översiktsbeskrivning av en dator

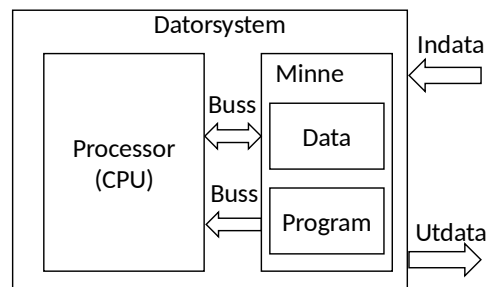
- Indata
 - Switchar, sensorer, kommunikationsmoduler
- Utdata
 - Lysdioder, reläer, video, etc.
- In och utdata i digital form
 - Binära tal
- Två huvuddelar
 - Minne lagrar data och program
 - Processor utför operationer
 - Bussar som skickar data/adress



- Minnet innehåller två typer av information
 - Instruktioner (vad som ska göras)
 - Datavärden (värden att räkna på)

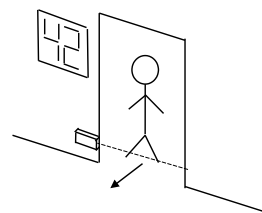
Principiell funktion

- Processorn läser instruktioner från minnet
 - Lagrade i binär form
 - Kallas oftast "Primärminne"
- Utför instruktionerna på värden från indata och minne
 - Kan lagra mellanresultat inuti processorn
- Slutligt resultat hamnar i minne och som utsignal



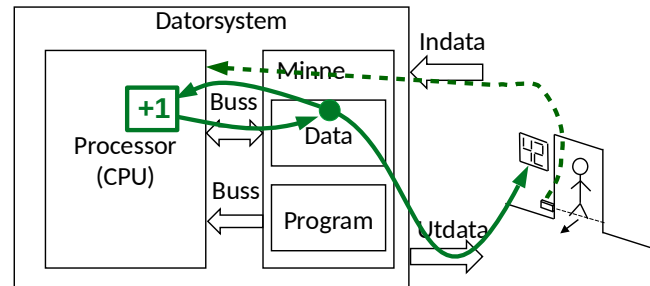
Litet exempel: Räkna studenter

- Jag vill räkna antal studenter som besöker mig på kontoret under en dag
 - Sätt sensor i dörren och en sifferdisplay
 - En dator får räkna
 - Ni som läst digitalteknik: ni vet hur man bygger detta med två räknare, 7-segmentsavkodare, enpulsare etc.
 - Detta byggsätt saknar flexibilitet när nya funktioner ska läggas till



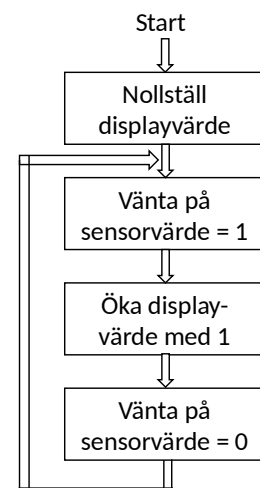
Litet exempel, lösning med dator

- Datorprogrammet ökar ett variabel-värde i minnet när någon passerar öppningen och uppdaterar display
- Displayvärde lagrat i minnet
- Öka värdet i minnet när sensor anger att någon passerar
- Skicka nya värdet till display



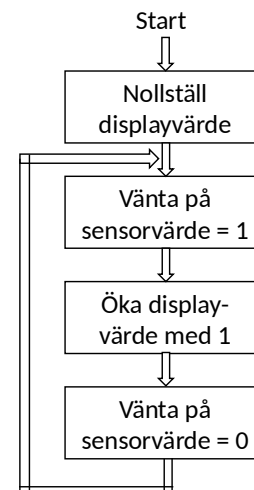
Första steget: definiera en algoritm

- Varje gång sensorvärdet ändras från 0 till 1 ska displayvärdet ökas med 1
- Dela upp i mindre steg
 - Initiering när strömmen slås på
 - Hitta sensorvärdesändring 0 till 1
 - Läs sensorvärde
 - Om sensorvärde inte 1 börja om med läsning
 - Öka displayvärde
 - Hitta sensorvärdesändring 1 till 0
 - Läs sensorvärde
 - Om sensorvärde inte 0 börja om med läsning
 - Börja om att leta efter sensor=1



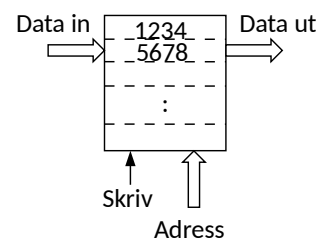
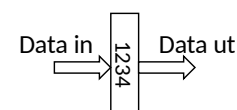
Funktioner/instruktioner som behövs

- Processorn kommer utföra instruktion efter instruktion (i en sekvens)
 - Jämför med att läsa ett recept
- Varje instruktion måste vara väldigt enkel (ska finnas hårdvara som utför detta senare)
- Tänk igenom och räkna upp olika aktiviteter
 - Sätta värde i minnet
 - Läsa värde från minnet
 - Skicka värde till display
 - Läsa av sensor (ljusmängd, t ex 0 eller 1 som värde)
 - Jämför värden
 - Börja om i sekvens från tidigare steg
 - Börja om i sekvens om speciellt värde fåtts



Typer av byggblock i processorn

- Register
 - Lagrar ett värde tillfälligt
 - En eller flera bitars värde
 - I exemplet: värdet på displayen lagras i ett register
- Minne
 - Större vektor av registervärden
 - Minnescell väljs via adress
 - Styrsignalen Skriv som väljer om data sparas eller läses



Storlekar på register och minnen

- Minsta enheten som kan lagras är en bit
 - Värde 0 eller 1
 - Oftast representerad som en spänning 0V eller 5V (eller liknande)
- Ofta grupperas bitar i större enheter
 - Nibble: 4 bitar (16 olika värden)
 - Byte: 8 bit (256 olika värden)
 - Halfword: 16 bitar (65536 olika värden)
 - Word: 32 bitar (4294967296 olika värden)
- Antal bitar in som adress bestämmer maximalt antal adresser i minnet
 - 0 indexerat (första värdet på adress 0)
 - $2^{\text{adressbitar}}$ stort
 - 10 bitar ger $2^{10} = 1024$ olika adresser från 0 till 1023
 - Ett dataord per adress (kan vara 1 eller flera byte)
 - I datasammanhang oftast 1 byte per adress

SI-prefix och datorer

- SI-prefix används för att förkorta notation av stora tal
 - $K = 10^3 = 1000$ (kilo)
 - $M = 10^6 = 1000000$ (mega)
 - $G = 10^9 = 1000000000$ (giga)
 - $T = 10^{12} = 1000000000000$ (tera)
 - $P = 10^{15} = 1000000000000000$ (peta)
- Stämmer mindre bra med minnesstorlekar (antag 1 byte/adress)
 - 10 bitar adress $\Rightarrow$ 1024 adresser $\sim$ 1 KByte (2.4 % fel)
 - 20 bitar adress $\Rightarrow$ 1048576 adresser $\sim$ 1 MByte (4.8 % fel)
- Ökande relativt fel med ökande storlek

Binära prefix

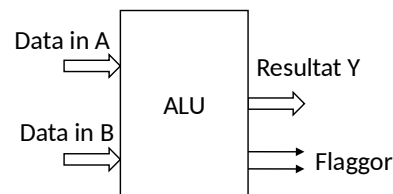
- Slarvig användning: 64KByte minne är 65536 byte stort
 - Används fortfarande för att beskriva primärminnets storlek
 - 16 GByte minnesmodul lagrar 17179869184 Byte
- Ny/tydligare notation: binära SI-prefix
 - Ki = 1024 (kibi)
 - Mi = 1024^2 (mebi)
 - Gi = 1024^3 (gibi)
 - Ti = 1024^4 (tebi)
 - Ex: 65536 Byte = $64 \cdot 1024 \Rightarrow 64$ KiByte
 - Används mest för sekundärminnen
 - Hårddisk, SSD, etc.
 - 1 TByte = $10^{12} \approx 931$ GiByte

Adress och bit positioner

- Alla positioner är noll-indexerade
 - Position längst till höger i ett binärt mönster är bitposition 0
 - Ett N-bitars register/värde har bitposition N-1 ned till 0
 - Adresser räknas från adress 0 och uppåt
 - 4 bitars adress $\Rightarrow 2^4 = 8$ kombinationer, adresser 0 till 7
 - En byte har bitpositioner räknat från 7 ned till 0
- Ofta beskrivs minnet som en array
 - Adress 0 ofta placerad längst upp
 - Adresser ökar nedåt
 - Matchar hur instruktioner i ett program körs en efter en med ökande adress

Typer av byggblock i processorn, forts.

- Aritmetikenhet för enklare beräkningar
 - Addition, subtraktion, and, or, etc.
 - Resultat analyseras (kontrollera om beräkning gav svar = 0, om minnessiffra från addition etc.)
 - Flaggor visar egenskaper om beräkningen/resultatet
- Styrsignaler väljer vilken funktion som ska beräknas.



$$Y = f(A,B)$$

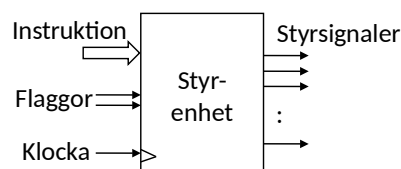
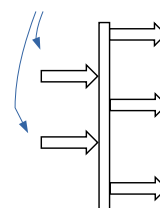
där
f kan vara AND, OR, XOR, NOT,
ADD,SUB,SHIFT,ROTATE,...

Flaggor anger egenskaper hos
resultat, t ex =0, minnessiffra, etc.

Typer av byggblock i processorn, forts.

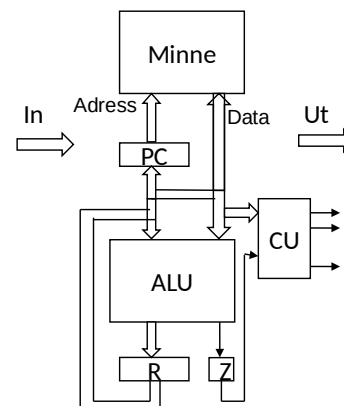
- Bussar
 - Skickar vidare värden från en block till många andra samtidigt
 - Bara ett värde åt gången på bussen
- Styrenhet
 - Avkoda vilken instruktion som ska utföras
 - Hämta nästa instruktion
 - Avkoda instruktion
 - Styr vilka värden som skickas vart
 - Påverkas ibland av flaggornas värde
 - Klocka styr hur ofta styrsignaler ändras

Endast en källa aktiv åt gången



Exekvering av program kräver minst

- Programräknare (PC)
 - Håller reda på position för aktuell instruktion i minnet
- Aritmetikenhet (ALU)
 - Beräknar addition etc.
 - Z: flagga = 1 om resultat = 0
- Tillfälliga register (R)
 - Kan vara flera stycken
- Styrenhet (CU)



Programmet lagras i minnet (exempel)

- Kallas en maskininstruktion
 - Binärt datamönster
 - Tolkas av styrenheten i processorn
- Lagras i en minnescell
 - I detta exempel: 32 bitar i en minnescell
- Två delar
 - Vilken typ av operation
 - Exempel: valt 4 bitar => 16 olika typer
 - Argument till operationen
 - Resten av bitarna: $32 - 4 = 28$ bitar argument

		Typ	Argument
		xxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
		4 bitar	28 bitar
Typkod	Förklaring		
0000 (0)	Ladda in argument i R		
0001 (1)	Addera argument till R		
0010 (2)	Jämför argument med R, Z=1 om R=argument, Z=0 annars		
0011 (3)	Sätt PC till argument om Z=0		
0100 (4)	Sätt PC till argument om Z=1		
0101 (5)	Sätt PC till argument		
0110 (6)	om argument = 2000 skicka R till display		
0110 (6)	om argument = 2100 hämta aktuellt displayvärde till R		
0110 (6)	om argument = 2200 hämta sensorvärde till R, R=1 om någon står i dörren, R=0 annars		
0111 (7)	läs värde till R från minne på adress i argumentet		
1000 (8)	skriv värde i R till minne på adress i argumentet		

Assemblerinstruktioner

- Svårt komma ihåg och läsa binärmönster
 - Använd så kallade mnemonics istället
 - Oftast förkortningar av funktionsnamnet

Typkod Assemblerinstruktion Förklaring

0000 (0)	mov R,#värde	Ladda in argument i R (MOVE)
0001 (1)	add R,#värde	Addera argument till R
0010 (2)	cmp R,#värde	Jämför R med argument, Z=1 om R=argument, Z=0 annars (CoMPare)
0011 (3)	bne adress	Sätt PC till argument om Z=0 (Branch Not Equal)
0100 (4)	beq adress	Sätt PC till argument om Z=1 (Branch Equal)
0101 (5)	b adress	Sätt PC till argument (Branch)
0110 (6)	bl 2000	om argument = 2000 skicka R till display (Branch and Link)
0110 (6)	bl 2100	om argument = 2100 hämta aktuellt displayvärde till R
0110 (6)	bl 2200	om argument = 2200 hämta sensorvärde till R, R=1 om någon står i dörren, R=0 annars

Assemblerinstruktioner, forts

- Olika processorfamiljer har olika mnemonics
 - Olika namn för samma funktion
 - Samma namn för olika funktioner
- Finns även pseudo-instruktioner i assembler
 - Styr assemblerns generering av maskininstruktioner
 - Kan ibland översätta en instruktion till en/flera andra

Z80	ARM	68000	x86_64
ld	ldr	move	mov
ld	str	move	mov
jr	b	bra	jmp
cp	cmp	cmp	cmp
add	add	add	add

Betydelse
Läs data från minne
Skriv data till minne
Hoppa i programmet
Jämför
Addera

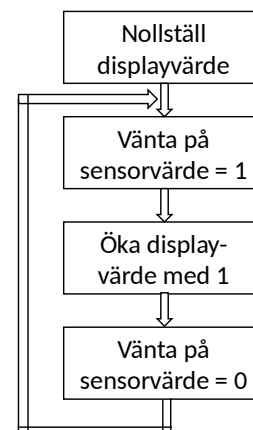
Pseudoinstruktion: Clr r0
Ska nollställa R0

Assembler översätter kanske till
xor r0,r0 ; r0 = r0 xor r0

Implementering i exempeldatorn

- Beskriv varje blocks funktion med hjälp av befintliga instruktioner

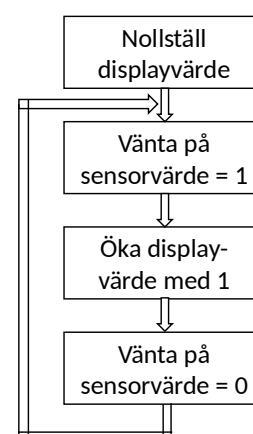
Adress	Maskinkod	Assembler-instruktion	Förklaring
0		mov R,#0	; Nollställ displayvärde
1		bl 2000	; sätt display till 00
2		bl 2200	; Hämta sensorvärde
3		cmp R,#1	; Står någon i dörren?
4		bne 2	; nej, gör om
5		bl 2100	; hämta displayvärdet
6		add R,#1	; öka displayvärdet
7		bl 2000	; visa det nya värdet
8		bl 2200	; Hämta sensorvärde
9		cmp R,#0	; Är dörren tom?
10		bne 8	; nej, kontrollera igen
11		b 2	; börja om



Implementering i exempeldatorn, forts.

- Översätt sedan till binär form
 - Dessa binära värden är vad som lagras i minnet

Adress	Maskinkod	Typ	Arg.	Assembler-instruktion	Förklaring
0	0	0		mov R,#0	; Nollställ displayvärde
1	6	2000		bl 2000	; sätt display till 00
2	6	2200		bl 2200	; Hämta sensorvärde
3	2	1		cmp R,#1	; Står någon i dörren?
4	3	2		bne 2	; nej, gör om
5	6	2100		bl 2100	; hämta displayvärdet
6	1	1		add R,#1	; öka displayvärdet
7	6	2000		bl 2000	; visa det nya värdet
8	6	2200		bl 2200	; Hämta sensorvärde
9	2	0		cmp R,#0	; Är dörren tom?
10	3	8		bne 8	; nej, kontrollera igen
11	5	2		b 2	; börja om



Assembler (hjälpprogram)

- Översätter assemblerinstruktioner (text) till maskinkod (binärdata)
 - Skriv en textfil med assemblerinstruktioner
- Håller även reda på adresser
 - Ange bara symboliska namn på platser i programmet (kallas label)

			Adress	Maskinkod
				Typ Arg.
wait0:	bl 2200	; Hämta sensorvärde	8	6 2200
	cmp R,#0	; Är dörren tom?	9	2 0
	bne wait0	; nej, kontrollera igen	10	3 8

↑
label



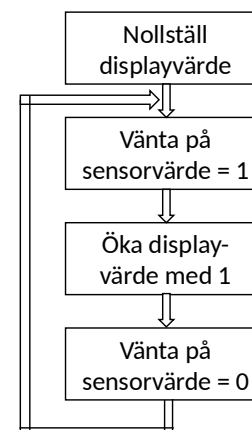
Assemblerversion av programmet

- Textbeskrivning, inklusive kommentarer
 - Label måste starta längst till vänster
 - Instruktioner måste ha ett mellanslag innan

```

start:  mov R,#0      ; Nollställ R
        bl 2000       ; sätt display till 00
wait1:  bl 2200       ; Hämta sensorvärde
        cmp R,#1      ; Står någon i dörren?
        bne wait1     ; nej, gör om
        bl 2100       ; hämta displayvärdet
        add R,#1      ; öka displayvärdet
        bl 2000       ; visa det nya värdet
wait0:  bl 2200       ; Hämta sensorvärde
        comp R,#0     ; Är dörren tom?
        bne wait0     ; nej, kontrollera igen
        b wait1       ; börja om

```



Ytterligare funktion: fördröjning

- Varje besökare kan ge flera pulser
 - Två ben, väska, kläder
- Lösning: vänta en stund efter display räknat upp
 - En loop i programmet som bara tar tid att utföra

Label	Assembler-instruktion	Förklaring
start:	mov R,#0 bl 2000	; Nollställ R ; sätt display till 00
wait1:	bl 2200 cmp R,#1 bne wait1 bl 2100 add R,#1 bl 2000	; Hämta sensorvärde ; Står någon i dörren? ; nej, gör om ; hämta displayvärdet ; öka displayvärdet ; visa det nya värdet
delay:	mov R,#0 add R,#1 cmp R,#10000 bne delay	; starta timer på 0 ; öka timer med 1 ; lämpligt antal klockcykler? ; inte tillräckligt många varv
wait0:	bl 2200 cmp R,#0 bne wait0 b wait1	; Hämta sensorvärde ; Är dörren tom? ; nej, kontrollera igen ; börja om

