

Introduktion till informationsteknologi (1DT051)

Binära tal, hexadecimala tal och två-komplement

september 2016

Uppsala universitet

karl.marklund@it.uu.se



Siffror och tal

I det decimala systemet har vi tio stycken siffror (symboler) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Med hjälp av dessa kan vi representera olika tal.

Siffran 1



1

hundra-tal

Siffran 2



2

tio-tal

Siffran 7



7

en-tal

I det decimala talsystemet tolkar vi detta som talet 127.

Addition av heltal

Hur går det till att addera positiva heltal?

127

56

+ ---

Addition av heltal

Hur går det till att addera positiva heltal?

$$\begin{array}{r} 1 \\ 127 \\ 56 \\ + --- \\ 3 \end{array}$$

Minnessiffra = överskjutande tiotal

tre 1-tal

Addition av heltal

Hur går det till att addera positiva heltal?

$$\begin{array}{r} \textcolor{violet}{1} \\ 127 \\ + 56 \\ \hline 183 \end{array}$$

åtta 10-tal

Addition av heltal

Hur går det till att addera positiva heltal?

$$\begin{array}{r} \textcolor{violet}{1} \\ 127 \\ + 56 \\ \hline 183 \end{array}$$

ett 100-tal

Decimala talsystemet (tiosystemet)

Ett positionssystem som baseras på talet 10 och därmed använder 10 olika siffror, det normala antalet fingrar, siffrorna 0–9. Sedan låter man siffrans position bestämma vilken 10-potens som siffran skall multipliceras med.

2 1 0 ← position

$$183_{10} = 3 \text{ ental} + 8 \text{ tiotal} + 1 \text{ hundratal}$$

talbasen anges
ofta som ett
nedsänkt **suffix**

$$183_{10} = 3 * 10^0 + 8 * 10^1 + 1 * 10^2 = 3 * 1 + 8 * 10 + 1 * 100$$

10-potenser

Tal i basen 3

Ett positionssystem som baseras på talet 3 och därmed använder 3 olika siffror: 0, 1, 2. Sedan låter man siffrans position bestämma vilken 3-potens som siffran skall multipliceras med.

2 1 0 ← position

$$212_3 = 2 * 3^0 + 1 * 3^1 + 2 * 3^2$$

3-potenser

$$= \begin{matrix} ental \\ 2 * 1 \end{matrix} + \begin{matrix} tretal \\ 1 * 3 \end{matrix} + \begin{matrix} niotal \\ 2 * 9 \end{matrix}$$
$$= 2 + 3 + 18$$
$$= 23_{10}$$

Vad är binära tal?

I det binära talsystemet finns endast två **siffror**: 1 (ett) och 0 (noll).

Med hjälp av dessa två siffror kan vi skapa olika binära **tal**, till exempel **101** och **1101**.

Varför är det praktiskt med binära tal?

Två tillstånd räcker:

- Ett påstående kan vara **sant/falskt**.
- Ett relä är **öppet/stängt**.
- En transistor ger **hög/låg** utspänning.
- Järnoxidskiktet på en liten yta av en diskett är **magnetiserad/omagnetiserad**.
- Metallsiktet på en liten yta av en CD-skiva reflekterar **mycket/litet** ljus.
- Ur en fiberoptisk kabel **lyser det/lyser det inte**.



En **bit** (**binary digit**) är den minsta representationen av ett tal eller logisk enhet inom digitaltekniken.

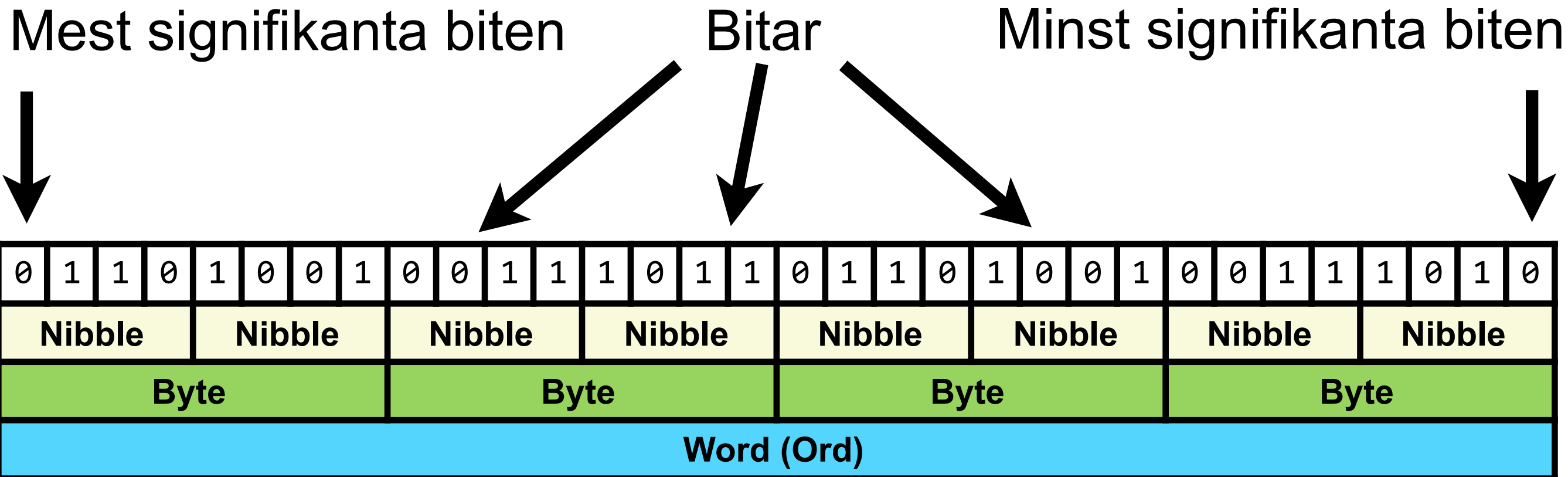
Varje bit kan ha två värden/tillstånd som fysiskt motsvaras på olika sätt, till exempel av/på i allmänhet, öppen/sluten brytare, släckt/tänd lampa, låg/hög nivå etc.

Bit, Nibble, Byte och Word (ord)

Vanliga namn på olika stora binära tal.

I datorsammanhang avser en **byte** storleken på den minsta adresserbara enheten.

Normalt består en byte av 8 bitar. En processor hanterar ofta flera bytes i taget, vanligen 2 eller 4 bytes, kallas **word** (ord). Med **MSB** avses **M**ost **S**ignifikant **B**it. Med **LSB** avses **L**east **S**ignifikant **B**it. En grupp om fyra bitar kallas i bland för **Nibble**.



Tal i basen 2 (binära tal)

Ett positionssystem som baseras på talet 2 och därmed använder 2 olika siffror: 0 & 1. Sedan låter man siffrans position bestämma vilken 2-potens som siffran skall multipliceras med.

2 1 0 ← position

$$101_2 = 1 * 2^0 +$$

$$0 * 2^1 +$$

$$1 * 2^2$$

$$= \begin{matrix} ental \\ 1 * 1 \end{matrix} + \begin{matrix} tvåtal \\ 0 * 2 \end{matrix} + \begin{matrix} fyrtal \\ 1 * 4 \end{matrix}$$


$$= 1 + 0 + 4$$

$$= 5_{10}$$

Konvertera från binärt till decimalt

$$11011_2 = ?_{10}$$

1	1	0	1	1	
4	3	2	1	0	<i>position</i>
16	8	4	2	1	<i>2-potens</i>



$$11011_2 = 16 + 8 + 2 + 1 = 27_{10}$$

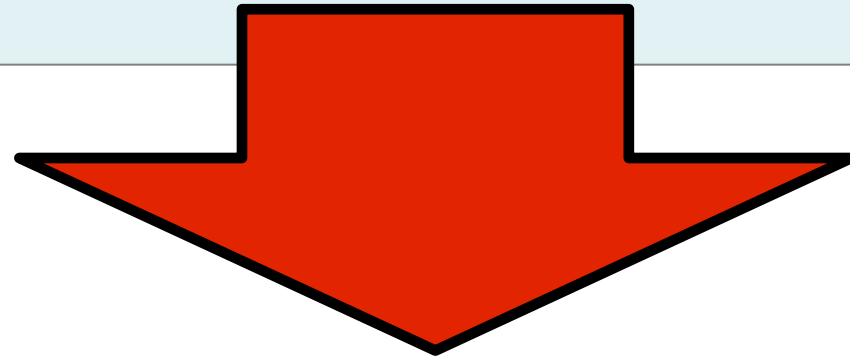
Konvertera från decimalt till binärt

$$21_{10} = ?_2$$

Konvertera från decimalt till binärt

$$21_{10} = ?_2$$

32	16	8	4	2	1	<i>2-potens</i>
0	1	0	1	0	1	



Använd "byggstenarna" 16, 4, 1
och "pussla" ihop till talet 21_{10}

$$10101_2 = (16+4+1)_{10} = 21_{10}$$

Jämna och udda tal

Vilka av följande binära tal är udda?



111₂

1010₂



10101₂

11010₂

10111111010₂



10111111011₂

För att ett binärt tal skall vara udda måste den **minst signifikanta** biten (längst till höger) vara 1.

Addition av binära tal

Ett 2-tal som
minnessiffra

Vi börjar med ensiffriga binära tal och får fyra fall.

	0	1	0	1	1	1
	0	0	1	1	1	1
+	-	+	-	+	-	+
	0	1	1	2	10	0

Hmm, om vi skall skriva
resultatet binärt får vi
endast använda
siffrorna **0** & **1**.

$$2_{10} = 10_2$$

0 ental

$$\begin{aligned} 2_{10} &= 10_2 \\ &= 0 \cdot 2^0 + 1 \cdot 2^1 \\ &= 0 \cdot 1 + 1 \cdot 2 \\ &= 2_{10} \end{aligned}$$

Addition av binära tal

Vi prövar att addera två stycken tresiffriga binära tal.

	1	1	1	
	1	0	1	5_{10}
	0	1	1	3_{10}
+	-	-	-	
	1	0	0	8_{10}

Notera att resultatet kräver 4 bitar. Har vi ej möjlighet att lagra mer än 3 bitar finns risk för **overflow**.

Samband mellan minnessiffra och overflow

(1)

Vanligen menas med aritmetiskt **overflow** att resultatet av en beräkning inte får plats i det utrymme som används för att lagra resultatet, dvs max antal siffror som kan användas för att representera (eller lagra) ett tal.

När en addition utförs görs detta position för position, dvs först adderas det minst signifikanta siffrorna och sedan de näst minst signifikanta siffrorna osv. Om resultatet av en sådan addition överskrider gränsen för vad som i talbasen går att representera med en siffra (symbol) flyttas resultatet över till nästa position (till vänster) i form av en **minnessiffra**.

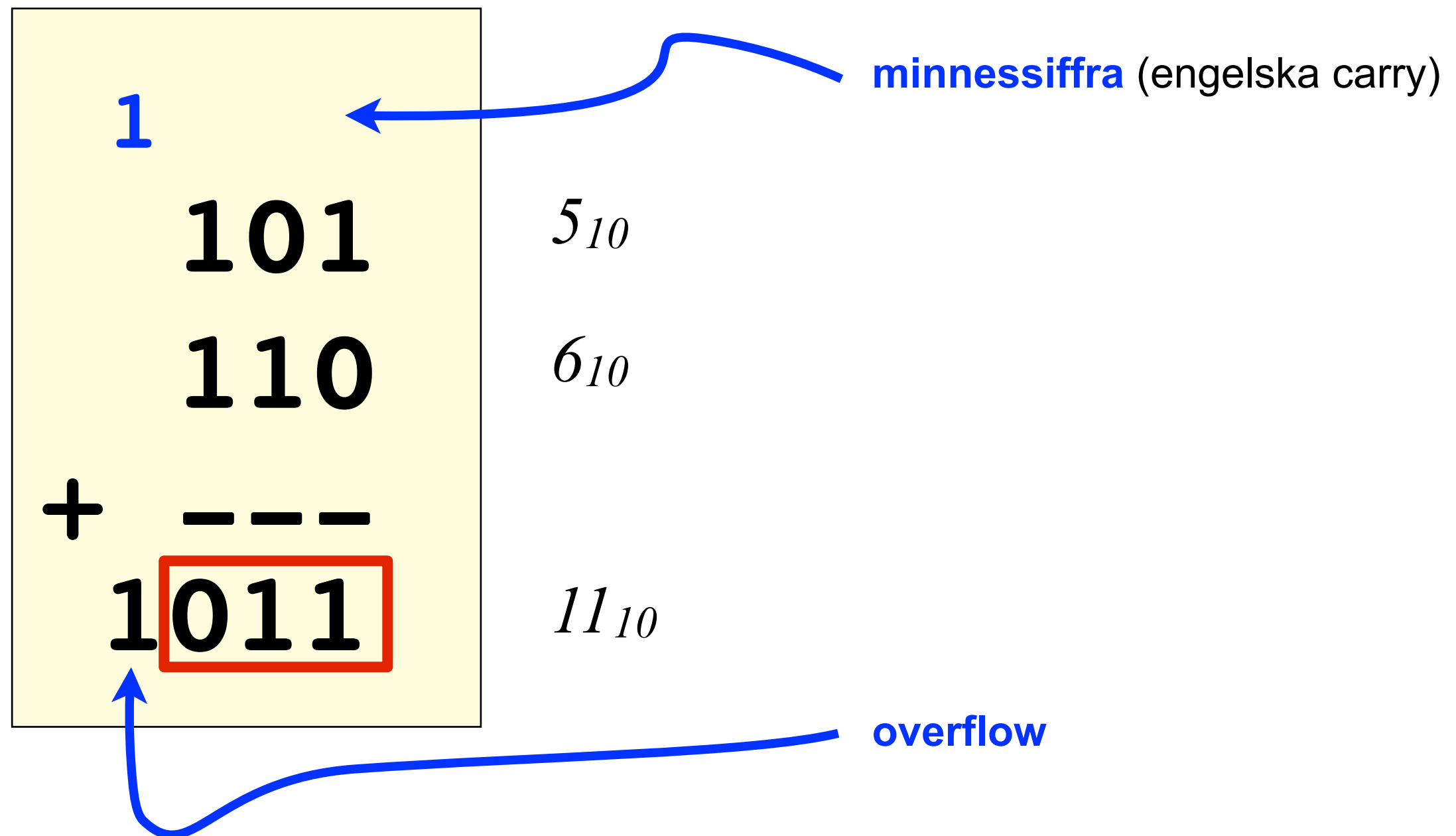
På sätt och vis kan minnessiffran ses som ett exempel på overflow vid addition av enstaka siffror.

Om det blir en minnessiffra i den mest signifikanta positionen som går att lagra (längst till vänster) får vi overflow då denna siffra inte får plats i det utrymme som finns att lagra resultatet.

Samband mellan minnessiffra och overflow

(2)

Om ett system endast kan hantera tal om 3 bitar och vi utför additionen $5_{10} + 6_{10} = 101_2 + 110_2 = 1011_2 = 11_{10}$ kan resultatet inte lagras på tre bitar på grund av overflow utan vi får $5_{10} + 6_{10} =_{\text{overflow}} 3_{10} = 011_2$.



Hexadecimala talsystemet

Ett positionssystem som baseras på talet 16 och därmed använder 16 olika siffror/symboler: 0-9, A, B, C, D, E, F.

1 0 ← position

4D₁₆

talbasen anges
ofta som ett
nedsänkt **suffix**

= 0x4D

= ?₁₀

ental

sextontal

= 13 * 16⁰ + 4 * 16¹

= 13 + 4 * 16

= 13 + 64

= 77₁₀

Hexadecimala tal
noteras ofta med ett
prefix i stället för suffix.
Vanligen används då
prefixet **0x**

Hex	Dec
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Konvertera från hexadecimalt till binärt

Det finns 16 olika ensiffriga hexadecimala tal:

$$(0-9, A-F)_{16} = (0-15)_{10}$$

Fråga: Hur många bitar behövs för att ange det största ensiffriga hexadecimala talet?

$$\begin{aligned} \mathbf{F}_{16} &= \mathbf{0xF} = \mathbf{15}_{10} \\ &= \mathbf{(8+4+2+1)_{10}} \\ &= \mathbf{1111}_2 \end{aligned}$$

Svar: Det räcker med fyra bitar!

Konvertera från hexadecimalt till binärt (forts)

Fråga: Om vi konverterar ett två siffror stort hexadecimalt tal till binär form, hur många bitar stor blir det binära talet?

Svar: 8 bitar.

Fråga: Hur konverterar vi det hexadecimala talet 0x4D till motsvarande binära tal?

Ett mycket praktiskt samband mellan de hexadecimala och binära talen gör att vi kan konvertera en hexadecimal siffra i taget (fyra bitar i taget, dvs en nibble i taget).

$$\begin{aligned} 0x4D_{16} &= \text{???? ????}_2 \\ &= 0100 \text{ ????}_2 \\ &= 0100 \ 1101_2 \\ &= 1+4+8+64_{10} \\ &= 77_{10} = (4 * 16 + 13)_{10} \end{aligned}$$

Hex	Dec	Bin
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Negativa binära tal

Vi har nu koll på hur vi kan representera tal binärt och hur positiva binära tal kan adderas.

Fråga: Vad händer om vi subtraherar två binära tal?

Svar: Resultatet kan bli negativt.

Fråga: Hur representeras negativa binära tal?

Sign-and-magnitude

Tal med tecken och storlek.

Ett alternativ är att låta den mest signifikanta biten (längst till vänster) representera talets tecken (0 positivt, 1 negativt).

Exempel:

$$+5_{10} = 0101_2$$

$$-5_{10} = 1101_2$$

$$+0_{10} = 0000_2$$

$$-0_{10} = 1000_2$$

När vi arbetar med binära tal med tecken måste vi bestämma oss för hur stora de är i antal bitar inklusive teckenbiten.

Notera att vi nu får två representationer av talet noll.

Two's complement

Två-komplement, ett system med mycket intressanta egenskaper.

I detta system gäller för alla storlekar på binära tal att ett tal med bara ettor alltid representerar minus 1.

För att göra det tydligt att talet skall tolkas med hjälp av tvåkomplement används =**signed**

$$1111111 \dots 1111_2 =_{\text{signed}} -1_{10}$$

Låt **Not (X)** beteckna den bitvisa **inversen** av ett binärt tal, dvs alla nollor bytta mot ettor och alla ettor bytta mot nollor.

$$X = 100101101 \dots 1011_2$$

$$\text{Not}(X) = 011010010 \dots 0100_2$$

Vi får ny följande samband:

$$X + \text{Not}(X) = 11111111 \dots 1111_2$$

$$X + \text{Not}(X) =_{\text{signed}} -1_{10}$$

Ofta utelämnas **signed** då det antas framgå av sammanhanget.

Two's complement (forts)


Konvertering från negativt decimaltal till binärt tal med två-komplement.

För tal tolkade med två-komplement gäller alltså:

$$\mathbf{X} + \mathbf{Not(X)} = -1_{10}$$

Då gäller även:

$$\mathbf{-X = Not(X) + 1}$$

$$\begin{aligned} -5 &= \mathbf{Not(5)} + 1 \\ &= \mathbf{Not(0101_2)} + 1 \\ &= \mathbf{1010_2} + 1 \\ &= \mathbf{1011_2} \end{aligned}$$


$5_{10} = 101_2$ men
vi behöver en bit
till för tecken.


Two's complement (forts)

Fler egenskaper när vi tolkar binära tal med hjälp av två-komplement.

För positiva tal (mest signifikanta biten = 0) kan vi summera alla 2-potenser på samma sätt som tidigare.

$$0101_2 =_{\text{signed}} 0 + 4 + 0 + 1 = +5_{10}$$

För negativa tal (mest signifikanta biten = 1) subtraherar vi den största 2-potensen och adderar övriga 2-potenser.

$$1011_2 =_{\text{signed}} -8 + 0 + 2 + 1 = -5_{10}$$


Detta gör det enkelt att med huvudräkning ta fram 2-komplementformen av ett tal.

$$\begin{aligned} -11_{10} &=_{\text{signed}} -16 + 5 = -2^4 + 2^1 + 2^0 \\ &= 10101_2 \end{aligned}$$

Subtraktion

Med hjälp av två-komplement kan vi subtrahera binära tal genom vanlig addition.

Vi testar att utföra additionen $5 + (-5)$.

$$5_{10} + (-5_{10}) =_{\text{signed}} 0101_2 + 1011_2 = 0000_2 = 0_{10}$$

För att representera talet 5 binärt behövs tre bitar:

$$5_{10} = (4+1)_{10} = 101_2$$

Talen i två-komplement måste alltså vara på fyra bitar:

$$+5_{10} = (4+1)_{10} = 0101_2$$

$$-5_{10} = (-8+2+1)_{10} = 1011_2$$

$$\begin{array}{r} 1111 \\ 0101 \\ 1011 \\ + \text{----} \\ 1\boxed{0000} \end{array}$$

Eftersom vi började med tal om fyra bitar (inklusive tecken) bryr vi oss bara om de fyra minst signifikanta bitarna av resultatet.

Subtraktion

När vi använder två-komplement är det viktigt att talen är av samma storlek

Vi testar att utföra additionen $13 + (-5)$.

$$13_{10} + (-5_{10}) =_{\text{signed}} 01101_2 + 11011_2 = +8_{10}$$

För att representera talet 13 binärt behövs fyra bitar:

$$13_{10} = (8+4+1)_{10} = 1101_2$$

För talet 5 behövs tre bitar:

$$5_{10} = (4+1)_{10} = 101_2$$

, men det behöver vara lika stort som 13_{10} , dvs fyra bitar:

$$5_{10} = (4+1)_{10} = 0101_2$$

Talen i två-komplement måste alltså vara på fem bitar:

$$+13_{10} = (8+4+1)_{10} = 01101_2$$

$$\begin{aligned} -5_{10} &= (-16+8+2+1)_{10} = 11011_2 \\ &= \text{Not}(00101)+1 = 11011_2 \end{aligned}$$

$$\begin{array}{r} 11111 \\ 01101 \\ 11011 \\ + \text{-----} \\ 101000 \end{array}$$

Eftersom vi började med tal om fem bitar (inklusive tecken) bryr vi oss bara om de fem minst signifikanta bitarna av resultatet.

Två-komplement

Om talen representeras och tolkas med två-komplement kan samma metod användas för addition och subtraktion. Beräkningen som utförs är identiskt, det är endast **tolkningen** av in- och utdata som skiljer.

$$\begin{array}{r} \overset{1}{1} \overset{1}{1} \overset{1}{0} 1 \\ 1101 \\ 0101 \\ + \text{-----} \\ 1\boxed{0010} \end{array}$$

unsigned, 5 bitar

13_{10}

5_{10}

18_{10}

signed, 4 bitar

-3_{10}

5_{10}

2_{10}

Kretsen beräknar **additionen** $1101_2 + 0101_2 = 13_{10} + 5_{10} = 18_{10} = 10010_2$. Notera att i detta fall, om vi bara kan lagra tal om fyra bitar får vi **overflow** och $1101_2 + 0101_2 = 13_{10} + 5_{10} = 2_{10} = 0010_2$.

Kretsen beräknar **subtraktionen** $1101_2 + 0101_2 =_{\text{signed}} -3_{10} + 5_{10} = 2_{10} = 0010_2$. Notera att i detta fall tolkar vi både indata och utdata (resultatet) som fyra bitars tal med två-komplement.

Two's complement (forts)

Praktiskt specialfall när vi tolkar tal med hjälp av två-komplement.

Låt oss försöka tolka följande stora (?) binära tal med hjälp av två-komplement.

$$\underline{11111} \underline{10101}_2 =_{\text{signed}} ?$$

$$= \boxed{-2^9 + 2^8} + 2^7 + \dots + 2^4 + 2^2 + 2^0$$

$$-2^9 + 2^8 = -256 = -2^8$$

$$= \boxed{-2^8 + 2^7} + \dots + 2^4 + 2^2 + 2^0$$

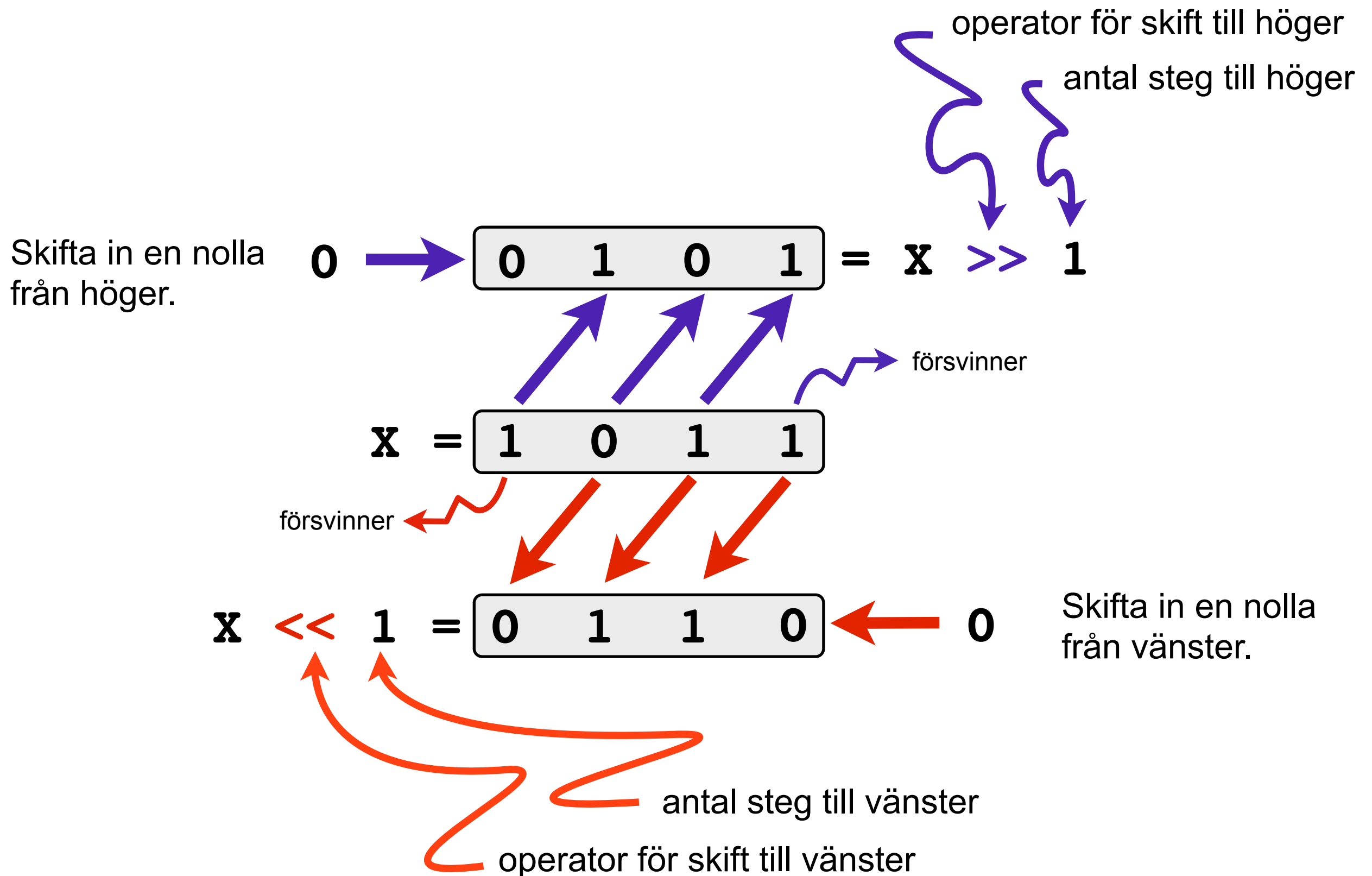
$$-2^8 + 2^7 = -128 = -2^7$$

Detta gör att vi kan "dra ihop" alla sammanhängande ettor från vänster.

$$= \underline{-2^4} + 2^2 + 2^0 = -16 + 4 + 1 = -11_{10}$$

Aritmetiskt skift

Binära tal kan skiftas till höger och till vänster.



Multiplikation och aritmetiskt skift av binära tal

Vi passar på att notera ett viktigt samband mellan aritmetiskt skift och multiplikation.

$$X = 0011_2 = 3_{10}$$

Skifta **ett** steg åt vänster

$$X \ll 1 = \underline{0011}_2 \ll 1 = \underline{0110}_2 = 6_{10} = 2 * X$$

Skifta **två** steg åt vänster

$$X \ll 2 = \underline{0011}_2 \ll 2 = \underline{1100}_2 = 12_{10} = 2 * 2 * X$$

Skifta **tre** steg åt vänster

$$X \ll 3 = \underline{0011}_2 \ll 3 = \underline{11000}_2 = 24 = 2 * 2 * 2 * X$$

Allmänt samband mellan skift åt vänster och multiplikation av binära tal.

$$X_2 * (2^n) = X_2 \ll n$$

Notera risk för overflow.

Allmänt om Multiplikation och aritmetiskt skift

Allmänt samband mellan skift åt vänster i alla talbaser.

$$\mathbf{X_b * (b^n) = X_b << n}$$









$$(3 * 8)_{10} = (3 * 2^3)_{10} = 000\underline{11}_2 << 3 = \underline{11000}_2 = 24_{10}$$

$$(375 * 10)_{10} = (375 * 10^1)_{10} = 00\underline{375}_{10} << 1 = \underline{03750}_{10}$$





$$(375 * 100)_{10} = (375 * 10^2)_{10} = 00\underline{375}_{10} << 2 = \underline{37500}_{10}$$

Övningsuppgifter

Under Allmän information i Studentportalen hittar du övningsuppgifter om binära och hexadecimala tal i filarean **Övrigt material** ➤ **1DT051_2016_Exercises_binary_and_hexadecimal_numbers.pdf**

-  Kursens startsida
-  Deltagarlista
-  Kursplan med kurslitteratur
-  Biblioteket
-  Mina grupper
-  Lärarlista
-  Kursschemat
-  Framsteg

▼ ALLMÄN INFORMATION


-  Allmän kursinformation
-  Kurslitteratur och läsanvisningar
-  Föreläsningsanteckningar
-  Övrigt material

Introduktion till informationsteknologi, 10 hp

Kurskod: 1DT051, Anmälningsskod: DT051, 67%, DAG, NML, vecka: 35 - 42 Termin: HT 2016

ÖVRIGT MATERIAL

Sökväg: Huvudmapp

 Ladda ner

☐ Dölj tomma mappar

	Namn	Storlek	Uppdaterad
	Huvudmapp		2016-08-31 av Karl Marklund
<input type="checkbox"/>	 Digitalteknik (Logisim)	9	2015-09-13 02:02 av Karl Marklund
<input type="checkbox"/>	 Python	1	2015-09-17 09:08 av Karl Marklund
<input type="checkbox"/>	 1DT051_2016_Exercises_binary_and_hexadecimal_numbers.pdf	87 kb	2016-08-31 18:50 av Karl Marklund
<input type="checkbox"/>	 1DT051_2016_Workshop_Problemlösning_och_Algoritmer.pdf	4905 kb	2016-08-31 18:50 av Karl Marklund