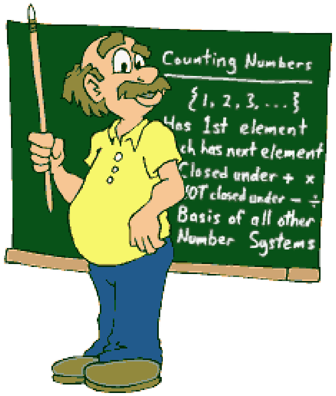
AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

MSYS

Microcontroller Systems

Lektion 9
Aritmetiske og logiske instruktioner



Version: 26-9-2017, Henning Hargaard

Binær addition

$$
\begin{array}{lr}
X & 190 \\
Y & +\,141 \\
\hline
X+Y & 331
\end{array}
\qquad
\begin{array}{ccccccccc}
 & & & 1 & 1 & 1 & 1 & & \\
 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
+ & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1
\end{array}
\qquad\qquad
\begin{array}{lr}
X & 173 \\
Y & +\;44 \\
\hline
X+Y & 217
\end{array}
\qquad
\begin{array}{ccccccccc}
 & & 1 & & & 1 & 1 & & \\
 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
+ & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
\hline
 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1
\end{array}
$$

Figure 2–1 Examples of decimal and corresponding binary additions.

Binær subtraktion

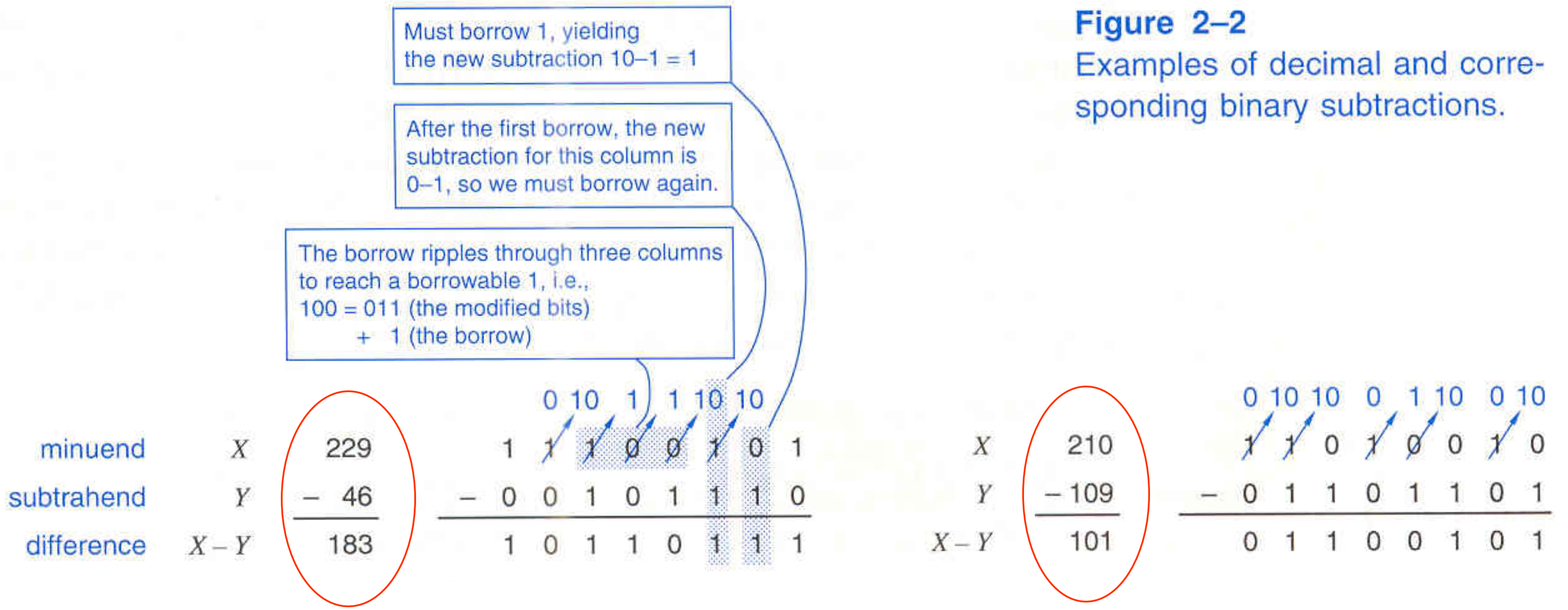
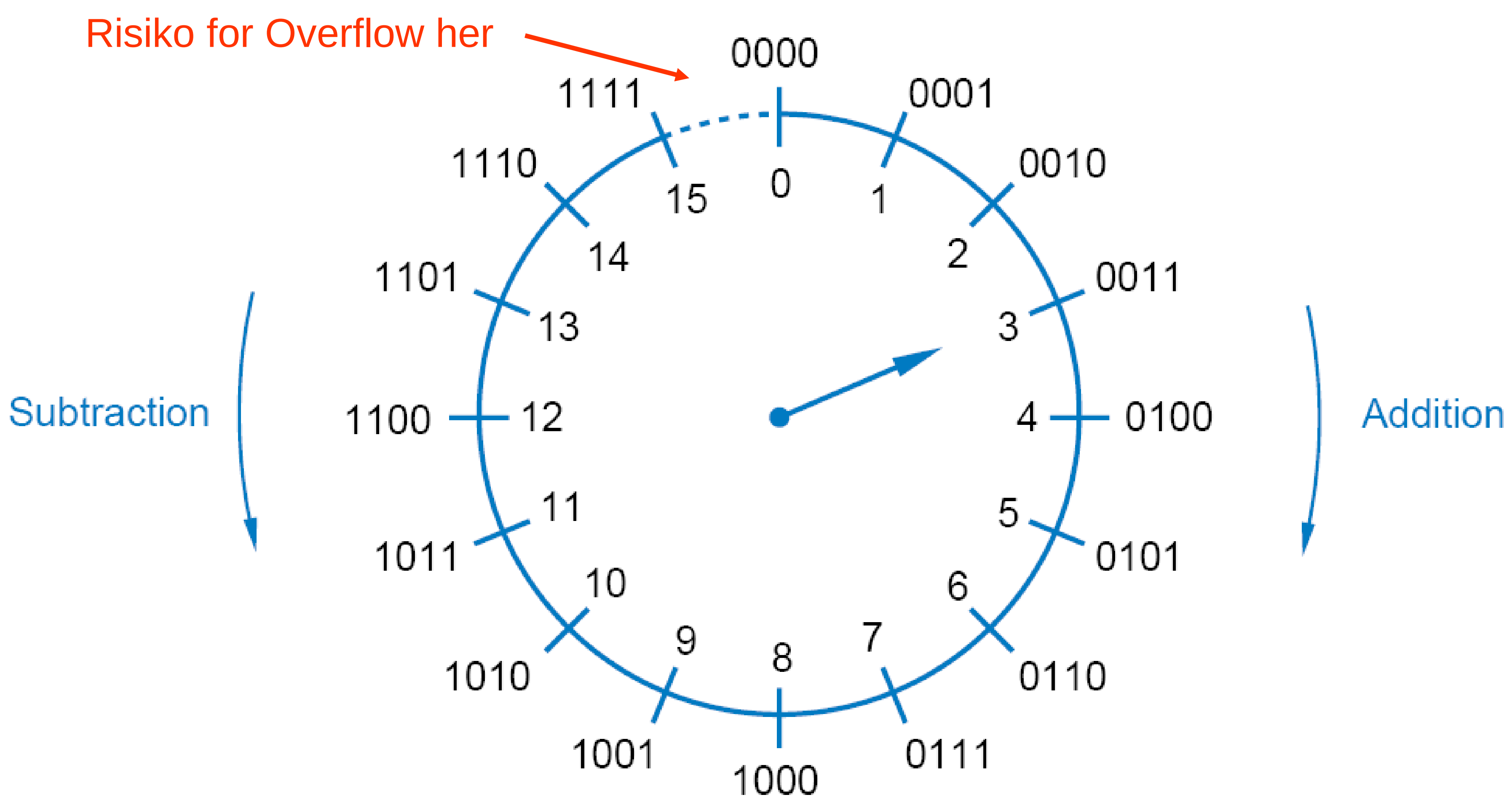


Figure 2–2
Examples of decimal and corresponding binary subtractions.

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

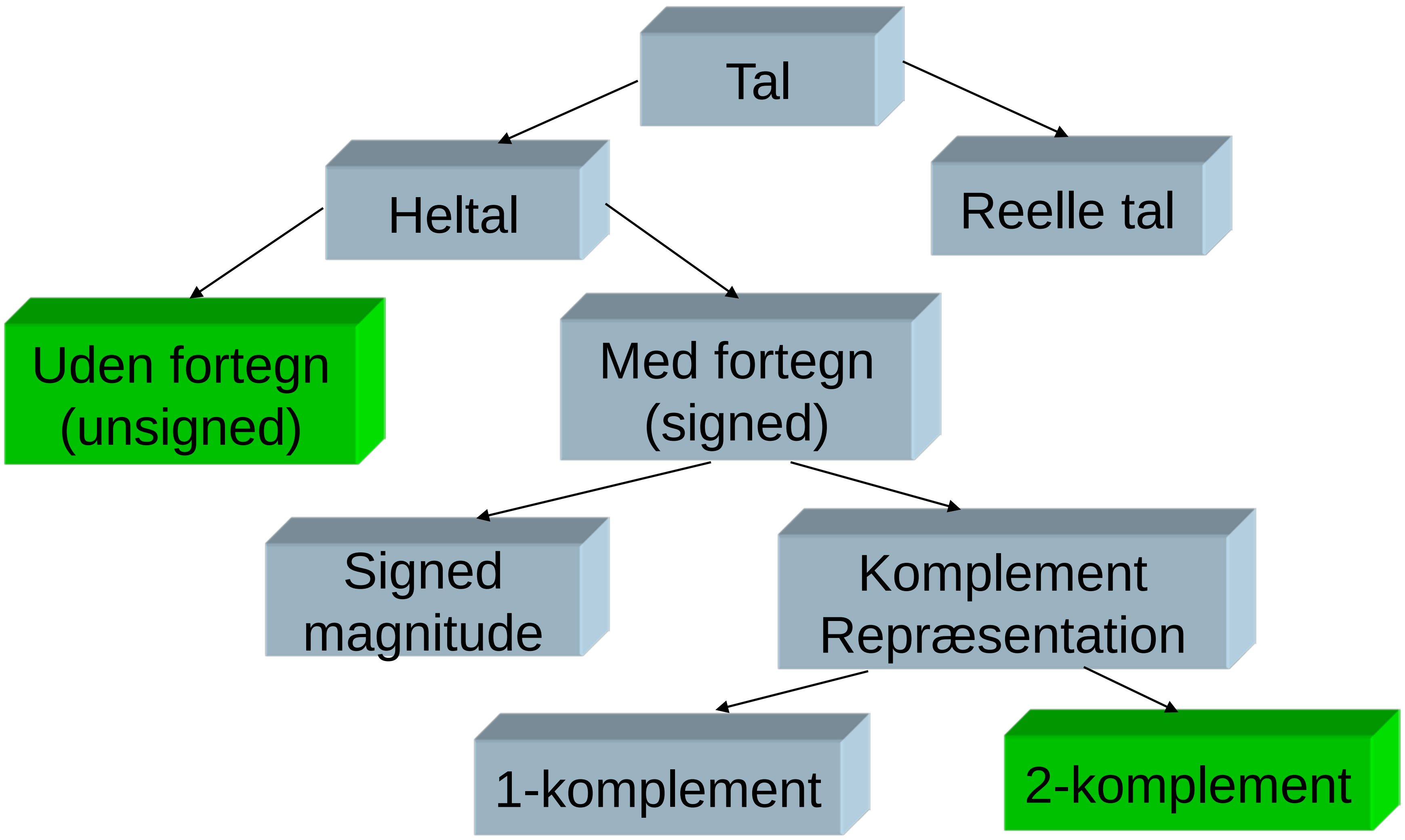
Unsigned addition / subtraktion



Nemt med 16 fingre :

C	1 1 0 0		1	1	0	0
X	$1\,9\,B\,9_{16}$		1	9	11	9
Y	$+\;C\,7\,E\,6_{16}$	+	12	7	14	6
$X+Y$	$E\,1\,9\,F_{16}$		14	17	25	15
			14	16+1	16+9	15
			E	1	9	F

Tal-repræsentationer



Signed magnitude

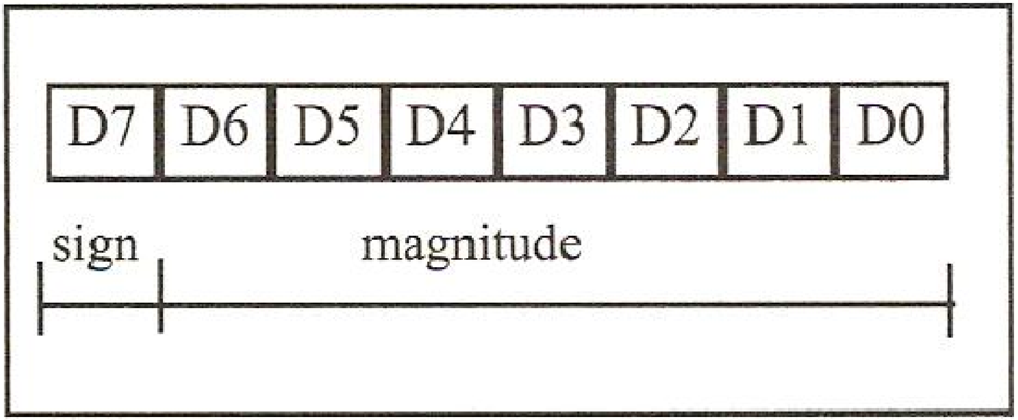
- Vi anvender typisk MSB til at repræsentere fortegnet.

$$01010101_2 = +85_{10} \qquad 11010101_2 = -85_{10}$$

$$01111111_2 = +127_{10} \qquad 11111111_2 = -127_{10}$$

$$00000000_2 = +0_{10} \qquad 10000000_2 = -0_{10}$$

- Bemærk, at vi har 2 værdier, der begge repræsenterer 0 !



Dannelse af 1-komplement

$$17_{10} = 00010001_2$$
$$\Downarrow$$
$$11101110_2 = -17_{10}$$

$$119_{10} = 01110111_2$$
$$\Downarrow$$
$$10001000_2 = -119_{10}$$

$$-99_{10} = 10011100_2$$
$$\Downarrow$$
$$01100011_2 = 99_{10}$$

$$-127_{10} = 10000000_2$$
$$\Downarrow$$
$$01111111_2 = 127_{10}$$

$$0_{10} = 00000000_2 \text{ (positive zero)}$$
$$\Downarrow$$
$$11111111_2 = 0_{10} \text{ (negative zero)}$$

- Bemærk : 2 værdier for 0 !

Dannelse af 2-komplement

$$17_{10} = 00010001_2$$
$\Downarrow$ complement bits
$$\begin{array}{r} 11101110 \\ +1 \\ \hline 11101111_2 \end{array} = -17_{10}$$

$$-99_{10} = 10011101_2$$
$\Downarrow$ complement bits
$$\begin{array}{r} 01100010 \\ +1 \\ \hline 01100011_2 \end{array} = 99_{10}$$

$$119_{10} = 01110111_2$$
$\Downarrow$ complement bits
$$\begin{array}{r} 10001000 \\ +1 \\ \hline 10001001_2 \end{array} = -119_{10}$$

$$-127_{10} = 10000001_2$$
$\Downarrow$ complement bits
$$\begin{array}{r} 01111110 \\ +1 \\ \hline 01111111_2 \end{array} = 127_{10}$$

$$0_{10} = 00000000_2$$
$\Downarrow$ complement bits
$$\begin{array}{r} 11111111 \\ +1 \\ \hline 1\,00000000_2 \end{array} = 0_{10}$$

$$-128_{10} = 10000000_2$$
$\Downarrow$ complement bits
$$\begin{array}{r} 01111111 \\ +1 \\ \hline 10000000_2 \end{array} = -128_{10}$$

- Regel : Inverter alle bit og læg en til.

Nummer systemer

Table 2-6 Decimal and 4-bit numbers.

Decimal	Two's Complement	Ones' Complement	Signed Magnitude
–8	1000	—	—
–7	1001	1000	1111
–6	1010	1001	1110
–5	1011	1010	1101
–4	1100	1011	1100
–3	1101	1100	1011
–2	1110	1101	1010
–1	1111	1110	1001
0	0000	1111 or 0000	1000 or 0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	0101	0101
6	0110	0110	0110
7	0111	0111	0111

2-komplement addition

$$\begin{array}{rr} +3 & 0011 \\ +\ \ +4 & +\ \ 0100 \\ \hline +7 & 0111 \end{array} \qquad \begin{array}{rr} -2 & 1110 \\ +\ \ -6 & +\ \ 1010 \\ \hline -8 & 1\,1000 \end{array}$$

$$\begin{array}{rr} +6 & 0110 \\ +\ \ -3 & +\ \ 1101 \\ \hline +3 & 1\,0011 \end{array} \qquad \begin{array}{rr} +4 & 0100 \\ +\ \ -7 & +\ \ 1001 \\ \hline -3 & 1101 \end{array}$$

- Bemærk : Samme hardware / instruktioner som ved unsigned behandling.
  Det er blot vores måde at kode tal på, der er anderledes !

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

”2-komplement – uret”

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

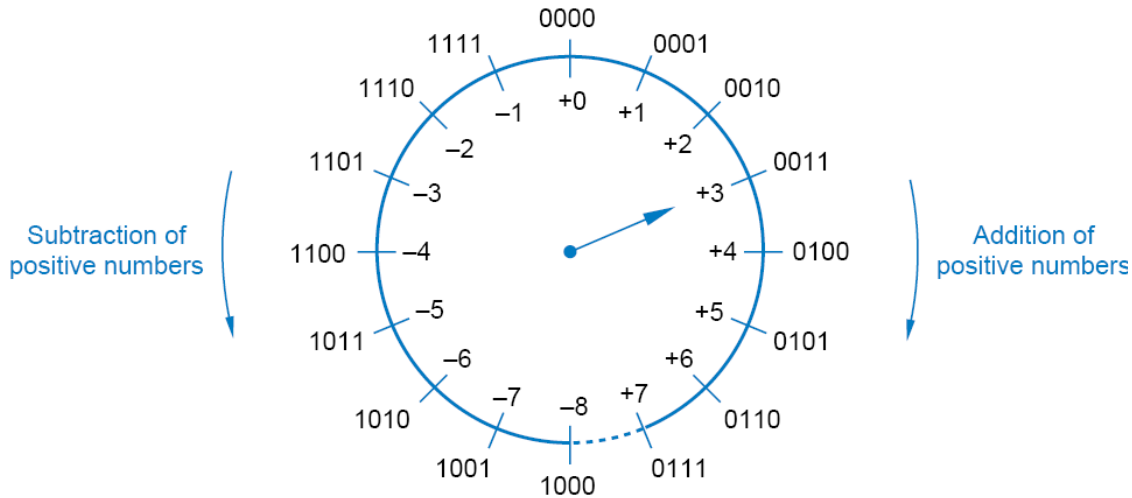
2-komplement Overflow

$$\begin{array}{rr} & -3 \\ + & -6 \\ \hline & -9 \end{array} \qquad \begin{array}{rr} & 1101 \\ + & 1010 \\ \hline & 1\,0111 = +7 \end{array}$$

$$\begin{array}{rr} & +5 \\ + & +6 \\ \hline & +11 \end{array} \qquad \begin{array}{rr} & 0101 \\ + & 0110 \\ \hline & 1011 = -5 \end{array}$$

$$\begin{array}{rr} & -8 \\ + & -8 \\ \hline & -16 \end{array} \qquad \begin{array}{rr} & 1000 \\ + & 1000 \\ \hline & 1\,0000 = +0 \end{array}$$

$$\begin{array}{rr} & +7 \\ + & +7 \\ \hline & +14 \end{array} \qquad \begin{array}{rr} & 0111 \\ + & 0111 \\ \hline & 1110 = -2 \end{array}$$



AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Test (”socrative.com”: Room = MSYS)

Hvordan gemmes tallet -5 (= minus 5) i ”signed magnitude” formatet ?

A: 0b00000101

B: 0b10000101

C: 0b11111010

D: 0b11111011

Test ("socrative.com": Room = MSYS)

Hvordan gemmes tallet -5 (= minus 5) i ”1’s complement” formatet ?

A: 0b00000101

B: 0b10000101

C: 0b11111010

D: 0b11111011

Test ("socrative.com": Room = MSYS)

Hvordan gemmes tallet -5 (= minus 5) i "2's complement" formatet ?

A: 0b00000101

B: 0b10000101

C: 0b11111010

D: 0b11111011

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Additions Instruktioner

```
ADD   Rd,Rr        ;Rd = Rd + Rr

ADC   Rd,Rr        ;Rd = Rd + Rr + C

ADIW  Rd:Rd,K      ;Rd+1:Rd = Rd+1:Rd + K
```

ADD (Add without Carry)

Description:

Adds two registers without the C Flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd + Rr$

	Syntax:	**Operands:**	**Program Counter:**
(i)	ADD Rd,Rr	$0 \le d \le 31, 0 \le r \le 31$	$PC \leftarrow PC + 1$

16-bit Opcode:

0000	11rd	dddd	rrrr

Example:

```
add    r1,r2     ; Add r2 to r1 (r1=r1+r2)
add    r28,r28   ; Add r28 to itself (r28=r28+r28)
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

ADC (Add with Carry)

Description:

Adds two registers and the contents of the C Flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd + Rr + C$

	Syntax:	**Operands:**	**Program Counter:**
(i)	ADC Rd,Rr	$0 \leq d \leq 31$, $0 \leq r \leq 31$	$PC \leftarrow PC + 1$

Example:

```
                ; Add R1:R0 to R3:R2
add   r2,r0     ; Add low byte
adc   r3,r1     ; Add with carry high byte
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Addition af 16 bit tal

OPGAVE: Skriv kode, der lægger to 16 bit tal sammen.

	15 8	7 0
	R21	R20
	15 8	7 0
+	R23	R22
	15 8	7 0
	R21	R20

Hint: Gør brug af instruktionerne ADD og ADC.

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Subtraktions Instruktioner

SUB	Rd,Rr	;Rd = Rd – Rr
SBC	Rd,Rr	;Rd = Rd - Rr - C
SUBI	Rd,K	;Rd = Rd - K
SBCI	Rd,K	;Rd = Rd - K – C
SBIW	Rd:Rd,K	;Rd+1:Rd = Rd+1:Rd - K

Vores microcontroller anvender denne metode for subtraktion (den ”lægger sammen for at trække fra”).

1. Danner 2-komplementet af tallet, der skal subtraheres.
2. Adderer de to tal.
3. Inverterer Carry-flaget.

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Multiplikation

Table 5-1: Multiplication Summary

Multiplication	Application	Byte1	Byte2	High byte of result	Low byte of result
MUL Rd, Rr	Unsigned numbers	Rd	Rr	R1	R0
MULS Rd, Rr	Signed numbers	Rd	Rr	R1	R0
MULSU Rd, Rr	Unsigned numbers with signed numbers	Rd	Rr	R1	R0

The following example multiplies 25H by 65H.

```
LDI    R23,0x25      ;load 25H to R23
LDI    R24,0x65      ;load 65H to R24
MUL    R23,R24       ;25H * 65H = E99 where
                     ;R1 = 0EH and R0 = 99H
```

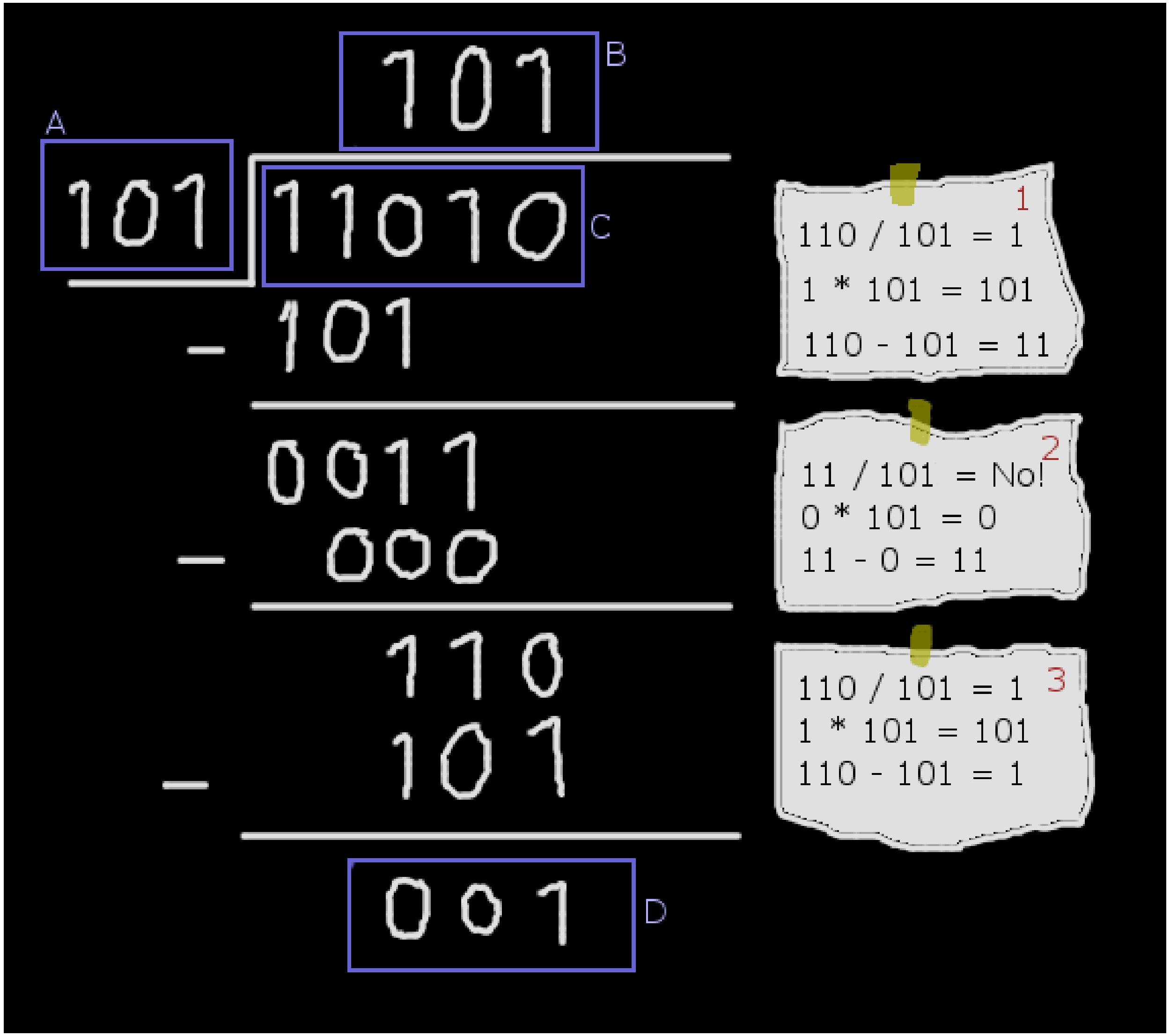
AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Eksempel på divisions-algoritme (s.167)

```
.DEF   NUM = R20
.DEF   DENOMINATOR = R21
.DEF   QUOTIENT = R22

       LDI    NUM,95               ;NUM = 95
       LDI    DENOMINATOR,10       ;DENOMINATOR = 10
       CLR    QUOTIENT             ;QUOTIENT = 0

L1:    INC    QUOTIENT
       SUB    NUM, DENOMINATOR
       BRCC   L1                   ;branch if C is zero

       DEC    QUOTIENT             ;once too many
       ADD    NUM, DENOMINATOR     ;add back to it

HERE: JMP HERE                     ;stay here forever
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

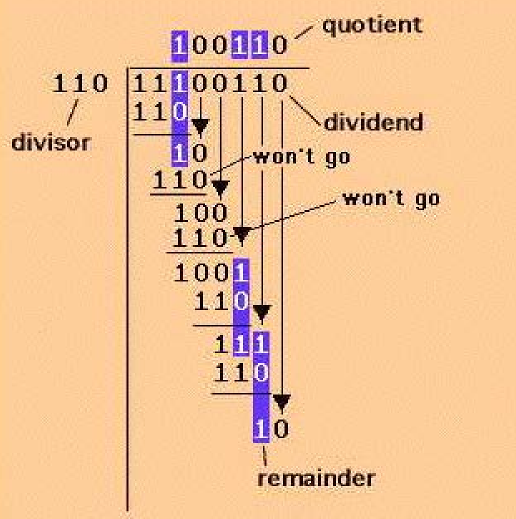
Binær division med shift / subtract



AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Binær division med shift / subtract

- **Set** quotient to 0
- Align leftmost digits in dividend and divisor
- **Repeat**
  - **If** that portion of the dividend above the divisor is greater than or equal to the divisor
    - **Then** subtract divisor from that portion of the dividend and
    - Concatentate 1 to the right hand end of the quotient
    - **Else** concatentate 0 to the right hand end of the quotient
  - Shift the divisor one place right
- **Until** dividend is less than the divisor
- quotient is correct, dividend is remainder
- **STOP**

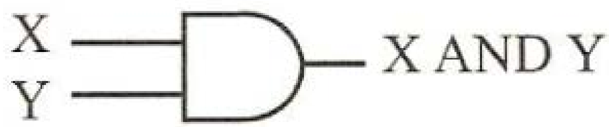


AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

AND Rd,Rr (Rd = Rd AND Rr)

Logical AND Function

Inputs		Output
X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1



Velegnet til at NULSTILLE bestemte bit.

Også muligt at AND'e med en konstant:
ANDI Rd,K ;rd = Rd AND K

AND eksempel

Example 5-18

Show the results of the following.

```
LDI     R20,0x35     ;R20 = 35H
ANDI    R20,0x0F     ;R20 = R20 AND 0FH (now R20 = 05)
```

Solution:

```
         35H    0011 0101
AND      0FH    0000 1111
         --------------
         05H    0000 0101          ;35H AND 0FH = 05H, Z = 0, N = 0
```

OR Rd,Rr (Rd = Rd OR Rr)

Logical OR Function

Inputs		Output
X	**Y**	**X OR Y**
0	0	0
0	1	1
1	0	1
1	1	1

Velegnet til at SÆTTE bestemte bit.

Også muligt at OR'e med en konstant:
ORI Rd,K ;rd = Rd OR K

OR eksempel

Example 5-19

(a) Show the results of the following:

```
LDI   R20,0x04            ;R20 = 04
ORI   R20,0x30            ;now R20 = 34H
```

(b) Assume that PB2 is used to control an outdoor light, and PB5 to control a light inside a building. Show how to turn "on" the outdoor light and turn "off" the inside one.

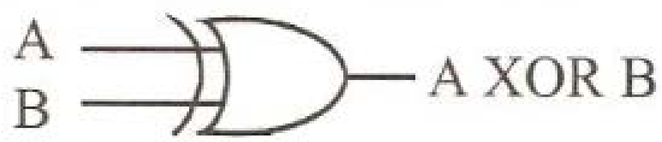
Solution:

```
(a)       04H     0000 0100
     OR   30H     0011 0000
          ------------------
          34H     0011 0100      04 OR 30 = 34H, Z = 0 and N = 0
(b)
     SBI  DDRB, 2              ;bit 2 of Port B is output
     SBI  DDRB, 5              ;bit 5 of Port B is output
     IN   R20, PORTB           ;move PORTB to R20.(Notice that we read
                               ;the value of PORTB instead of PINB
                               ;because we want to know the last value
                               ;of PORTB, not the value of the AVR
                               ;chip pins.)
     ORI  R20, 0b00000100      ;set bit 2 of R20 to one
     ANDI R20, 0b11011111      ;clear bit 5 of R20 to zero
     OUT  PORTB, R20           ;out R20 to PORTB

HERE: JMP HERE                 ;stop here
```

EOR Rd,Rr (Rd = Rd XOR Rr)

Logical XOR Function

Inputs		Output
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0



Velegnet til at INVERTERE bestemte bit.

EOR eksempel

Example 5-20

Show the results of the following:

```
LDI   R20, 0x54
LDI   R21, 0x78
EOR   R20, R21
```

Solution:

```
        54H    0101 0100
XOR     78H    0111 1000
        ----------------
        2CH    0010 1100          54H XOR 78H = 2CH, Z = 0, N = 0
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Test (”socrative.com”: Room = MSYS)

Hvordan kan man nulstille bit 7 i R16 (uden at ændre på resten af bittene) ?

A: LDI R17, 0b10000000
AND R16, R17

B: LDI R17, 0b01111111
EOR R16, R17

C: LDI R17, 0b10000000
OR R16, R17

D: LDI R17, 0b01111111
AND R16, R17

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Test ("socrative.com": Room = MSYS)

Hvordan kan man <u>sætte bit 7</u> i R16 (uden at ændre på resten af bittene) ?

A: LDI R17, 0b10000000
AND R16, R17

B: LDI R17, 0b01111111
EOR R16, R17

C: LDI R17, 0b10000000
OR R16, R17

D: LDI R17, 0b01111111
AND R16, R17

Test ("socrative.com": Room = MSYS)

Hvordan kan man invertere / toggle bit 7 i R16 (uden at ændre på resten af bittene) ?

A: LDI R17, 0b10000000
AND R16, R17

B: LDI R17, 0b10000000
EOR R16, R17

C: LDI R17, 0b10000000
OR R16, R17

D: LDI R17, 0b01111111
AND R16, R17

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

COM og NEG

COM (complement)

This instruction complements the contents of a register. The complement action changes the 0s to 1s, and the 1s to 0s. This is also called *1's complement*.

```
LDI     R20,0xAA       ;R20 = 0xAA
COM     R20            ;now R20 = 55H
```

Logical Inverter

Input	**Output**
X	**NOT X**
0	1
1	0

NEG (negate)

This instruction takes the 2's complement of a register. See Example 5-23.

Example 5-23

Find the 2's complement of the value 85H. Notice that 85H is –123.

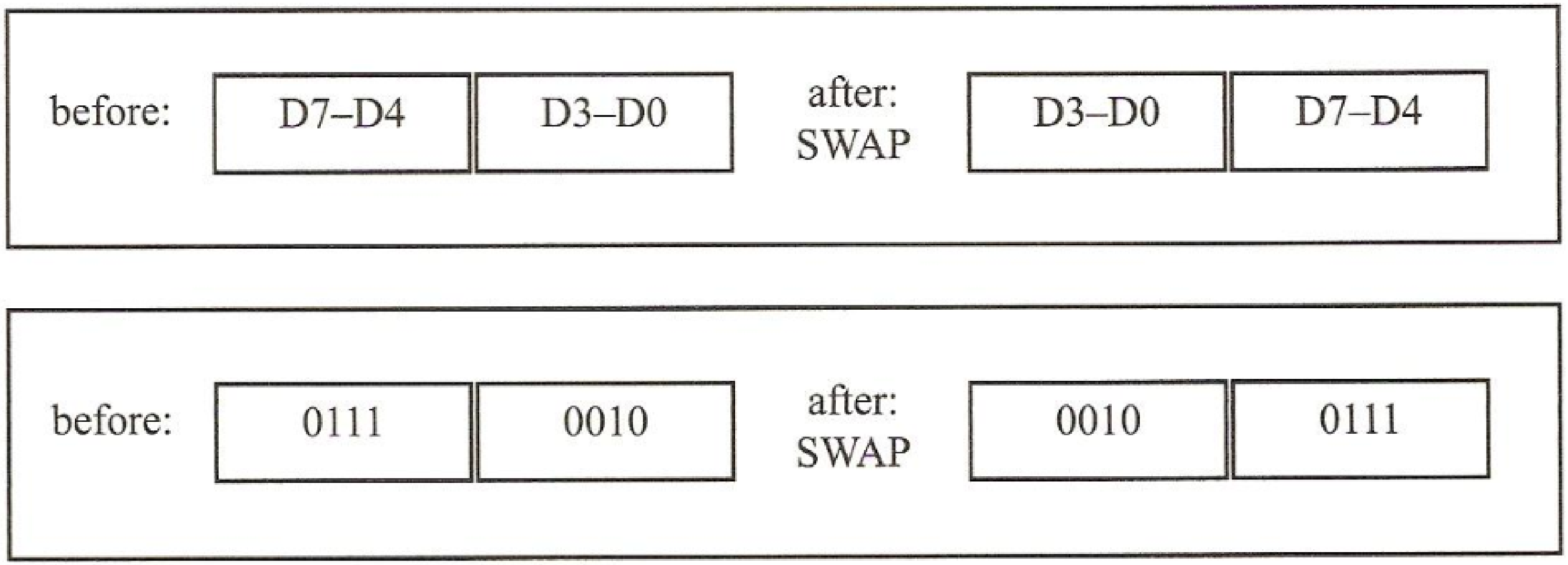
Solution:

```
     LDI    R21, 0x85                ;85H = 1000 0101
                                     ;1's = 0111 1010
                                             +    1
                                          ----------
     NEG    R21           ;2's comp       0111 1011 = 7BH
```


AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

SWAP Register

Eksempel: SWAP R16



Compare

CP	Rd,Rr	;Rd sammenlignes med Rr
CPI	Rd,K	;Rd sammenlignes med konstanten K

Table 5-2: AVR Compare Instructions

BREQ	Branch if equal	Branch if Z = 1
BRNE	Branch if not equal	Branch if Z = 0
BRSH	Branch if same or higher	Branch if C = 0
BRLO	Branch if lower	Branch if C = 1
BRLT	Branch if less than (signed)	Branch if S = 1
BRGE	Branch if greater than or equal (signed)	Branch if S = 0
BRVS	Branch if Overflow flag set	Branch if V = 1
BRVC	Branch if Overflow flag clear	Branch if V = 0

Conditional branch og flagene

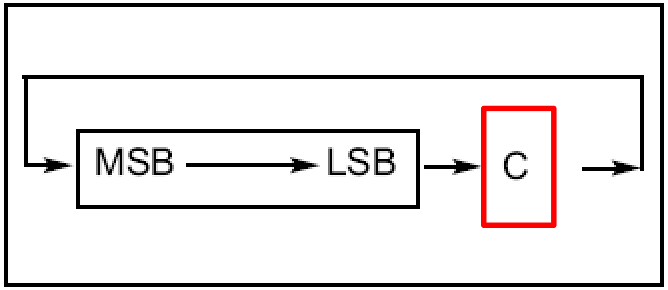
Test	Boolean	Mnemonic	Complementary	Boolean	Mnemonic	Comment
Rd > Rr	$Z \bullet (N \oplus V) = 0$	BRLT(1)	Rd ≤ Rr	$Z+(N \oplus V) = 1$	BRGE*	Signed
Rd ≥ Rr	$(N \oplus V) = 0$	BRGE	Rd < Rr	$(N \oplus V) = 1$	BRLT	Signed
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Signed
Rd ≤ Rr	$Z+(N \oplus V) = 1$	BRGE(1)	Rd > Rr	$Z \bullet (N \oplus V) = 0$	BRLT*	Signed
Rd < Rr	$(N \oplus V) = 1$	BRLT	Rd ≥ Rr	$(N \oplus V) = 0$	BRGE	Signed
Rd > Rr	C + Z = 0	BRLO(1)	Rd ≤ Rr	C + Z = 1	BRSH*	Unsigned
Rd ≥ Rr	C = 0	BRSH/BRCC	Rd < Rr	C = 1	BRLO/BRCS	Unsigned
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Unsigned
Rd ≤ Rr	C + Z = 1	BRSH(1)	Rd > Rr	C + Z = 0	BRLO*	Unsigned
Rd < Rr	C = 1	BRLO/BRCS	Rd ≥ Rr	C = 0	BRSH/BRCC	Unsigned
Carry	C = 1	BRCS	No carry	C = 0	BRCC	Simple
Negative	N = 1	BRMI	Positive	N = 0	BRPL	Simple
Overflow	V = 1	BRVS	No overflow	V = 0	BRVC	Simple
Zero	Z = 1	BREQ	Not zero	Z = 0	BRNE	Simple

Note: 1. Interchange Rd and Rr in the operation before the test, i.e., CP Rd,Rr → CP Rr,Rd

ROR instruction

```
ROR      Rd                    ;Rd  (only flags are set)
```

In ROR, as bits are rotated from left to right, the carry flag enters the MSB and the LSB exits to the carry flag. In other words, **in ROR the C is moved to the MSB, and the LSB is moved to the C.**



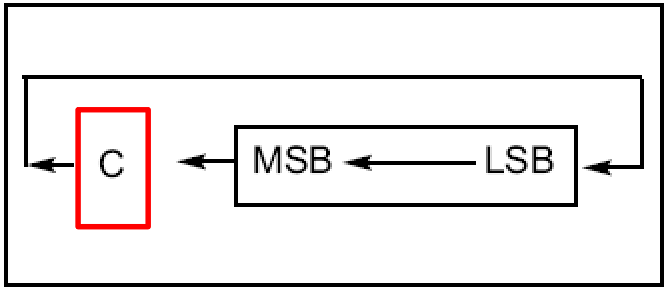
See what happens to 0010 0110 after running 3 ROR instructions:

```
CLC                      ;make C = 0 (carry is 0 )
LDI      R20 , 0x26      ;R20 = 0010 0110
ROR      R20             ;R20 = 0001 0011 C = 0
ROR      R20             ;R20 = 0000 1001 C = 1
ROR      R20             ;R20 = 1000 0100 C = 1
```

ROL instruction

ROL Rd ;Rd (only flags are set)

ROL. In ROL, as bits are shifted from right to left, the carry flag enters the LSB and the MSB exits to the carry flag. In other words, **in ROL the C is moved to the LSB, and the MSB is moved to the C.**

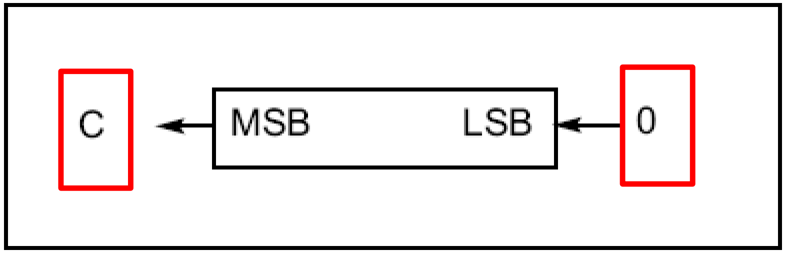



```
SEC              ;make C = 1 (carry is 0)
LDI R20,0x15     ;R20 = 0001 0101
ROL R20          ;R20 = 0010 1011 C = 0
ROL R20          ;R20 = 0101 0110 C = 0
ROL R20          ;R20 = 1010 1100 C = 0
ROL R20          ;R20 = 0101 1000 C = 1
```

LSL instruction

LSL   Rd   ;logical shift left

In LSL, as bits are shifted from right to left, 0 enters the LSB and the MSB exits to the carry flag. In other words, **in LSL 0 is moved to the LSB, and the MSB is moved to the C.**



this instruction multiplies content of the register by 2 assuming that after LSL the carry flag is not set.

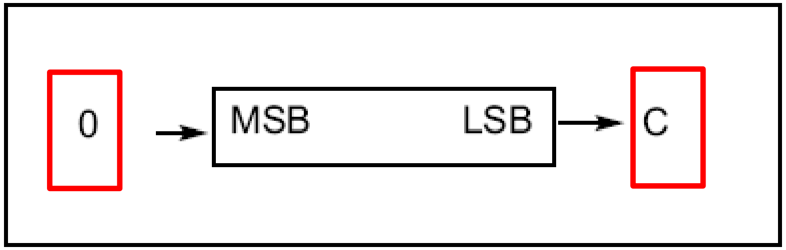
In the next code you can see what happens to 00100110 after running 3 LSL instructions.

```
CLC                ;make C = 0 (carry is 0 )
LDI R20 , 0x26     ;R20 = 0010 0110(38) c = 0
LSL R20            ;R20 = 0100 1100(74) C = 0
LSL R20            ;R20 = 1001 1000(148) C = 0
LSL R20            ;R20 = 0011 0000(98) C = 1 as C=1 and content of R20
                   ;is not multiplied by 2
```

LSR Instruction

LSR Rd ;Rd (only flags are set)

In LSR, as bits are shifted from left to right, 0 enters the MSB and the LSB exits to the carry flag. In other words, **in LSR 0 is moved to the MSB, and the LSB is moved to the C.**



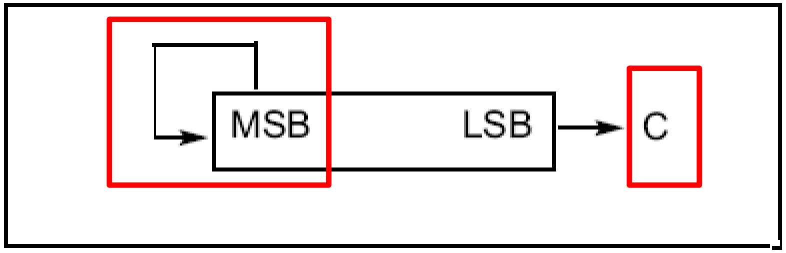
this instruction divides content of the register by 2 and carry flag contains the remainder of division.

In the next code you can see what happens to 0010 0110 after running 3 LSL instructions.

```
LDI R20,0x26     ;R20 = 0010 0110 (38)
LSR R20          ;R20 = 0001 0011 (19) C = 0
LSR R20          ;R20 = 0000 1001 (9) C = 1
LSR R20          ;R20 = 0000 0100 (4) C = 1
```

ASR Instruction

```
ASR     Rd                 ;Rd  (only flags are set)
```



ASR means *arithmetic shift right*. ASR instruction can divide signed number by 2. In LSR, as bits are shifted from left to right, MSB is held constant and the LSB exits to the carry flag. In other words **MSB is not changed but is copied to D6, D6 is moved to D5, D5 is moved to D4 and so on.**

In the next code you can see what happens to 0010 0110 after running 5 ASL instructions.

```
LDI   R20 , 0D60          ;R20 = 1101 0000(-48) c = 0
ASR  R20                  ;R20 = 1110 1000(-24) C = 0
ASR  R20                  ;R20 = 1111 0100(-12) C = 0
ASR  R20                  ;R20 = 1111 1010(-6) C = 0
ASR  R20                  ;R20 = 1111 1101(-3) C = 0
ASR  R20                  ;R20 = 1111 1110(-1) C = 1
```

Test (”socrative.com”: Room = MSYS)

Hvordan kan man (hurtigt) <u>dividere R16 med 2</u> ?
R16 indeholder et positivt tal.

A: ROR R16

B: ROL R16

C: LSR R16

D: LSL R16

E: DIV R16,2

Test ("socrative.com": Room = MSYS)

Hvordan kan man (hurtigt) gange R16 med 2 ?
R16 indeholder et positivt tal.

A: ROR R16

B: ROL R16

C: LSR R16

D: LSL R16

E: ASR R16

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

LAB5, del 1

MSB (bit 31)	Tal 1:		LSB (bit 0)
R19	R18	R17	R16

MSB (bit 31)	Tal 2:		LSB (bit 0)
R23	R22	R21	R20

MSB (bit 31)	Sum efter addition:		LSB (bit 0)
R19	R18	R17	R16

1 milliard + 2 milliarder = ?

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

LAB5, del 2

- Hvis trykknap SW7 aktiveres:
  Talværdien på PORTB inkrementeres.

- Hvis trykknap SW6 aktiveres:
  Talværdien på PORTB dekrementeres.

- Hvis trykknap SW5 aktiveres:
  Værdien på [LED7,LED6,LED5,LED4] "bytter plads med" [LED3,LED2,LED1,LED0].

- Hvis trykknap SW4 aktiveres:
  Alle lysdioderne skifter tilstand (fra tændt til slukket og omvendt).

- Hvis trykknap SW3 aktiveres:
  Talværdien på PORTB divideres med 8.

- Hvis trykknap SW2 aktiveres:
  Talværdien på PORTB divideres med 7 *(Hint: Se side 167 i lærebogen).*

- Hvis trykknap SW1 aktiveres:
  LED7 og LED0 slukkes, mens de andre LEDs skal være uændrede.

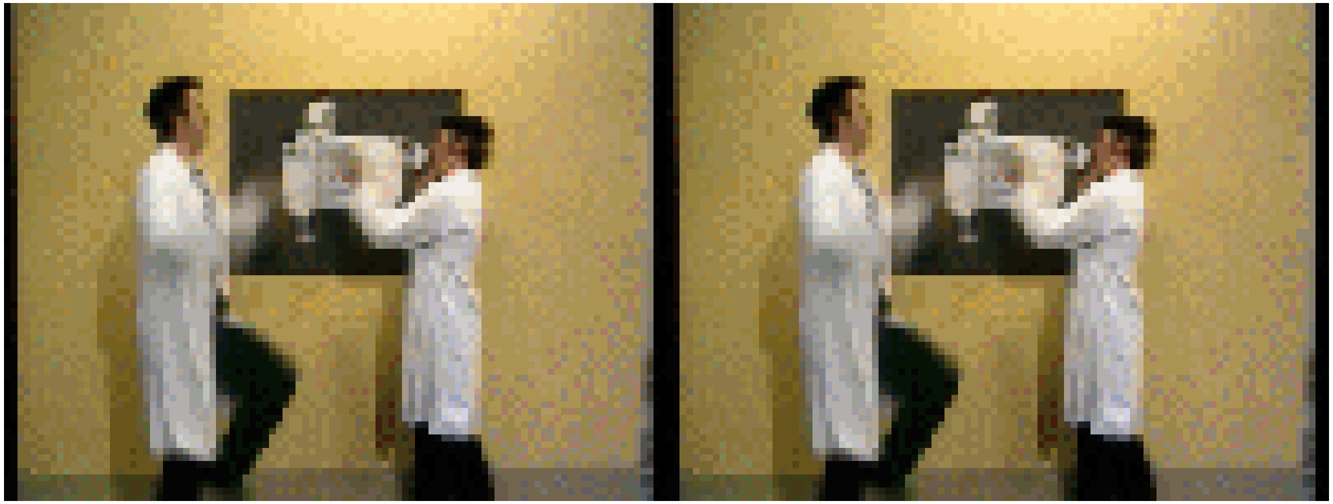
- Hvis trykknap SW0 aktiveres:
  LED7 og LED0 tændes, mens de andre LEDs skal være uændrede.

LAB5, del 3 (hvis tid)

```
;************ LED_OFF **********
;***** Slukker en LED på PB ****
;***** Bit nr.(0-7) i R20    ****
;*******************************
LED_OFF:
   LDI  R21,1            ;R21 = 0b00000001
   CPI  R20,0
   BREQ KLAR1            ;Hop, hvis LED nr. = 0
IGEN1:
   LSL  R21              ;Venstre-skift R21
   DEC  R20              ;ialt "LED nr." pladser
   BRNE IGEN1
KLAR1:
   COM  R21              ;Inverter "masken"
   IN   R20,PINB         ;Aflæs alle LEDs
   AND  R20,R21          ;- lav bitvis AND
   OUT  PORTB,R20        ;- og skriv ud til LEDs igen
   RET
;*******************************

;************ LED_ON ***********
;***** Taender en LED på PB ****
;***** Bit nr.(0-7) i R20    ****
;*******************************
LED_ON:

   ;<-------- Skriv den manglende kode her

   RET
;*******************************
```


AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Slut på lektion 9



AARHUS UNIVERSITY
SCHOOL OF ENGINEERING