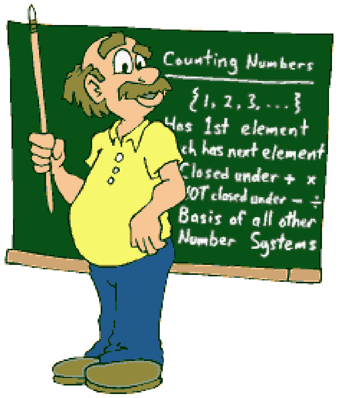
AARHUS
UNIVERSITY

SCHOOL OF ENGINEERING

MSYS

Microcontroller Systems

Lektion 6: Status register og delays



Version: 7-9-2017, Henning Hargaard

Ubetinget "langt" Jump (JMP)

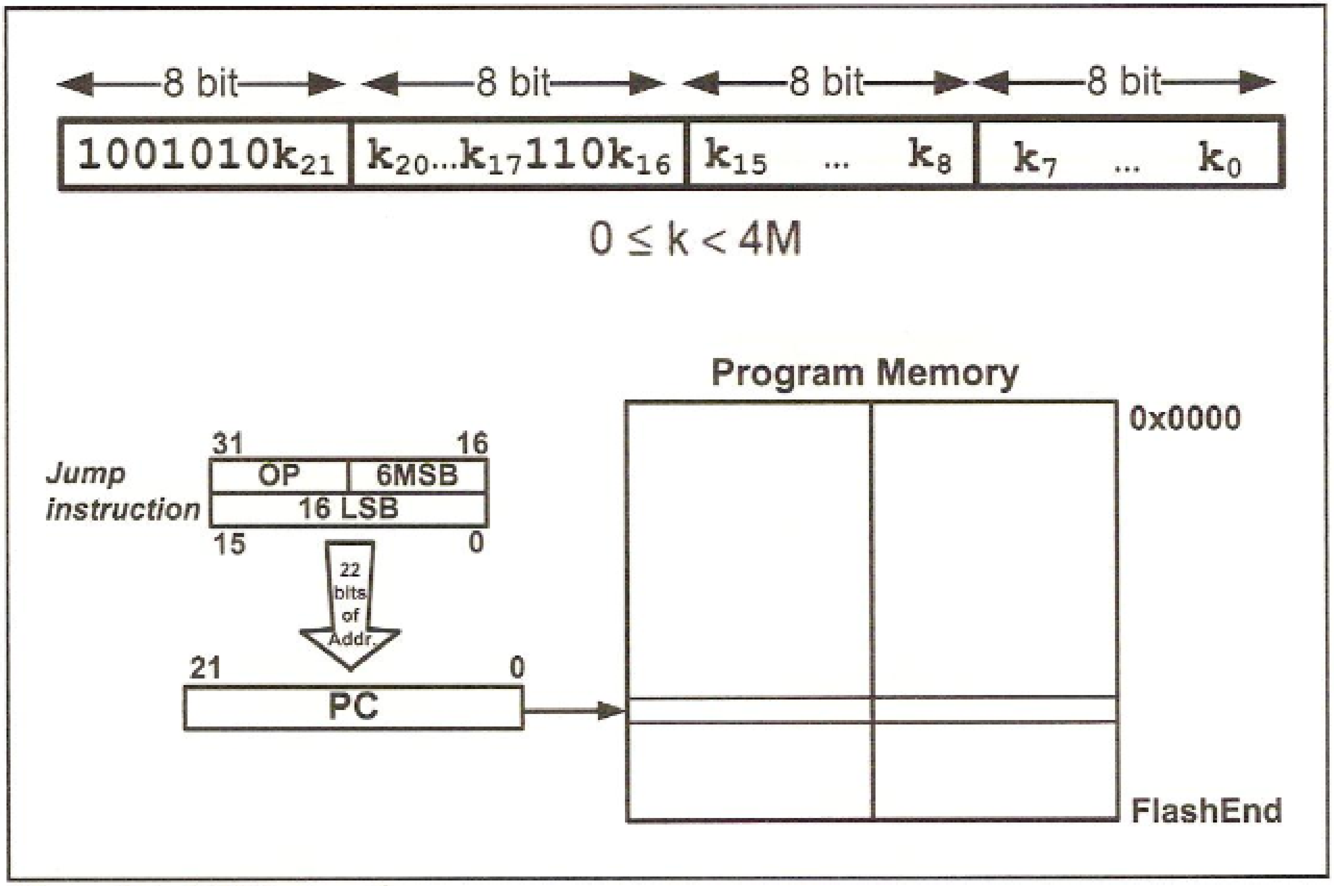


Figure 3-4. JMP Instruction

JMP IGEN

Ubetinget ”kort” Jump (RJMP)

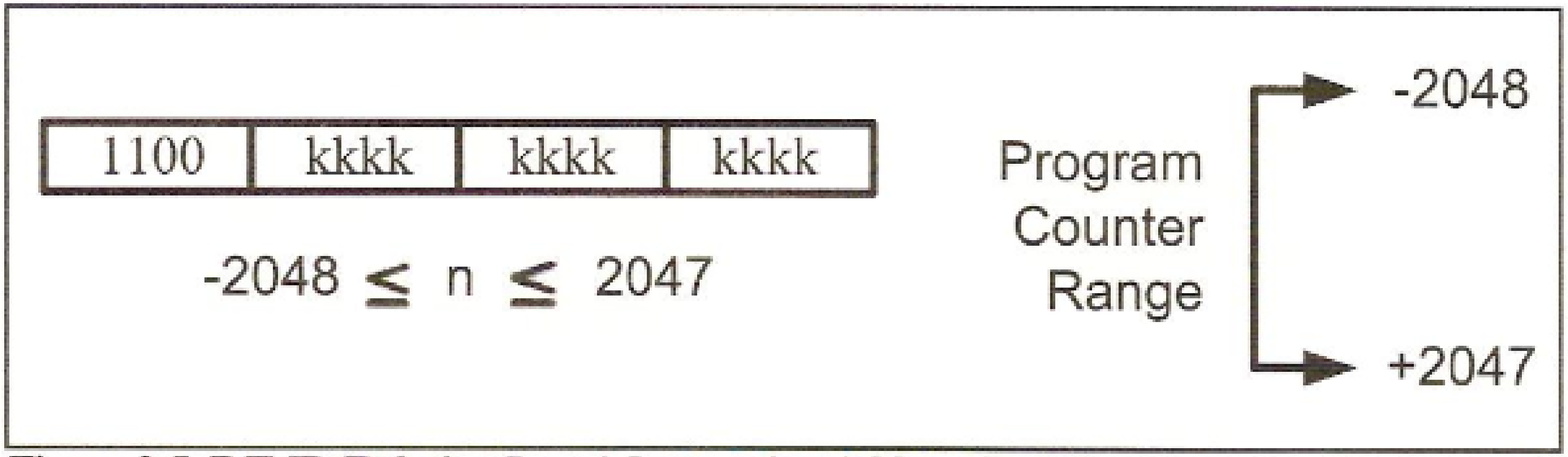


Figure 3-5. RJMP (Relative Jump) Instruction Address Range

```
RJMP  IGEN
```

Indirekte Jump (IJMP)

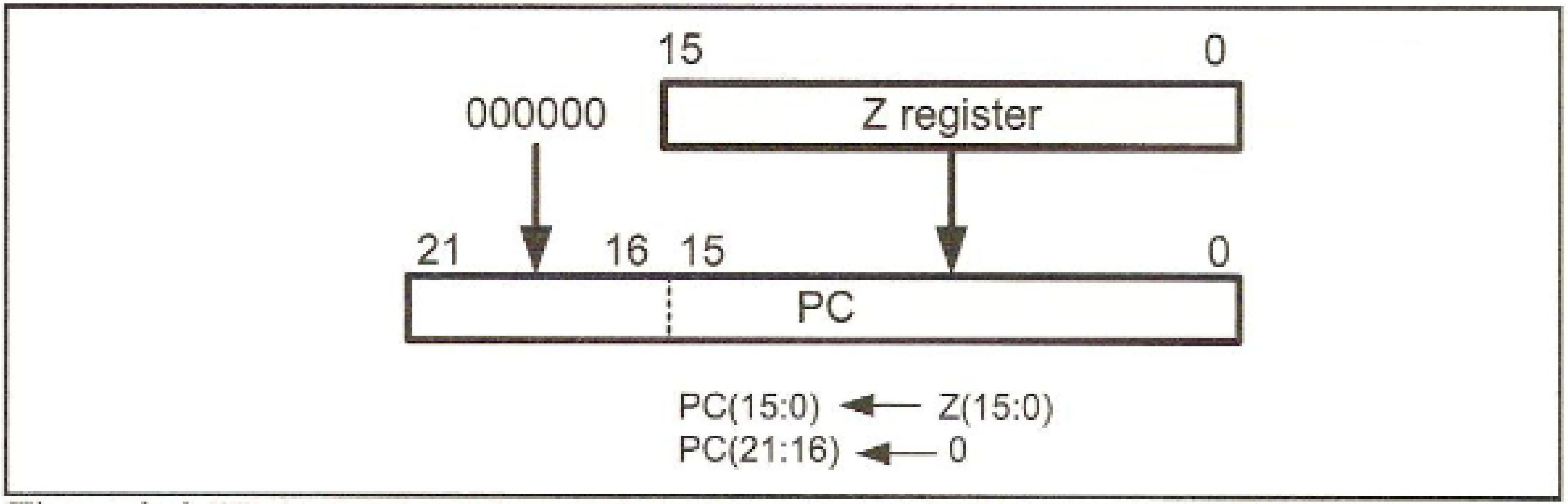


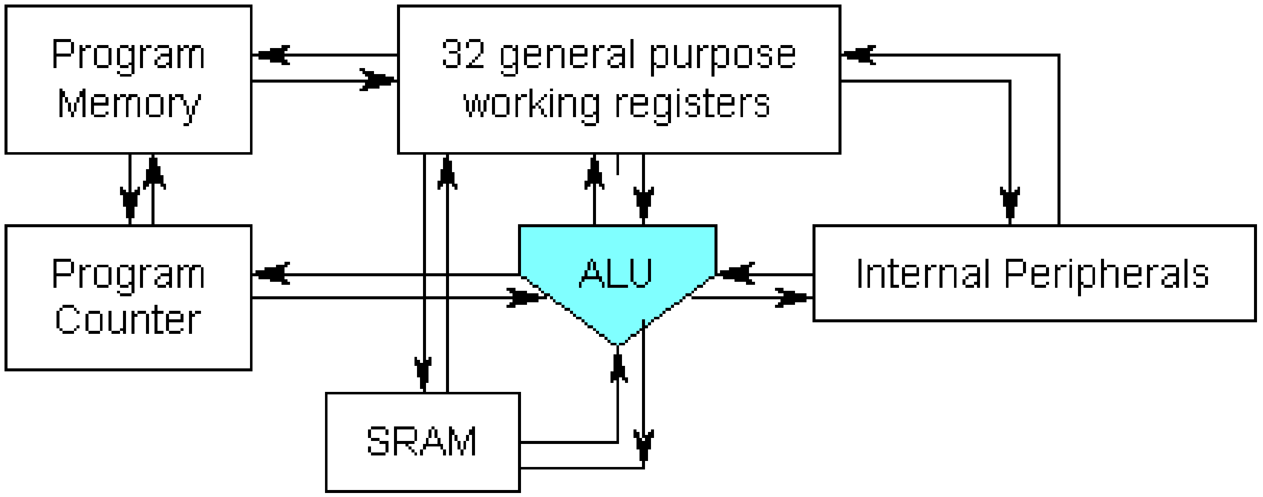
Figure 3-6. IJMP (Indirect Jump) Instruction Target Address

Anvendes sjældent !

```
LDI    R30,0xA5 ;Z register er fysisk det samme
LDI    R31,0x07  ;som R31 og R30 kombineret
IJMP             ;Hopper til 0x07A5 (= PC)
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

ALU

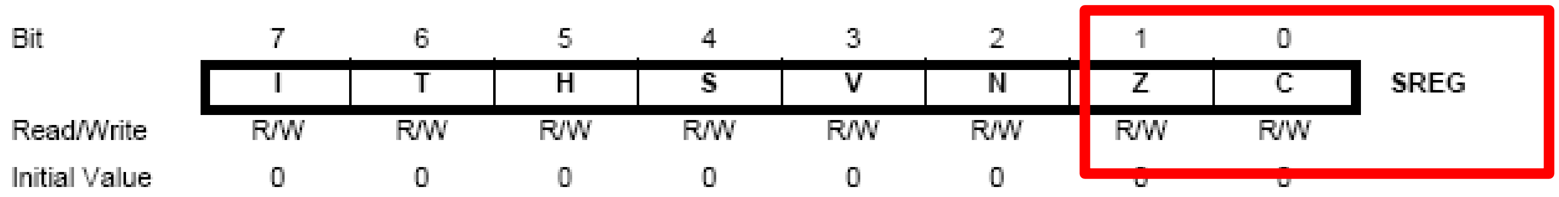


ALU'en er mikrocontrollerens "regnemaskine".

Hver gang vi udfører **en instruktion, der involverer ALU'en**, påvirkes nogen meget vigtige bits i status registeret (SREG) !

Hver bit i SREG kaldes "et **flag**".

Status register SREG



Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- I-flag : Global interrupt enable (CLI og SEI).
- T-flag : Bit copy mellemlager (BLD og BST).
- H-flag : ”Half carry” sættes ved ”mente” fra bit 3 til bit 4 i en beregning. Bruges sjældent.
- S-flag : ”Sign” kan anvendes ved beregning på negative tal. Afhænger af N- og V-flaget.
- V-flag : ”2-komplement overflow” kan anvendes ved beregning på negative tal.
- N-flag : ”Negative” indikerer et negativt resultat.
- **Z-flag : ”Zero”** sættes, hvis resultatet bliver 0.
- **C-flag : ”Carry”** sættes, hvis resultatet giver ”mente”.

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Hvilke instruktioner bruger ALU'en ?

- Det er ikke alle instruktioner, der bruger regnemaskinen (ALU'en).
- Eksempelvis vil **MOV R12,R16** og **LDI R20,7** ikke påvirke statusregisteret (SREG).
- Eksempler på instruktioner, der påvirker SREG-flagene er :

INC R17

ADD R16,R17

DEC R4

Her påvirkes status-flagene

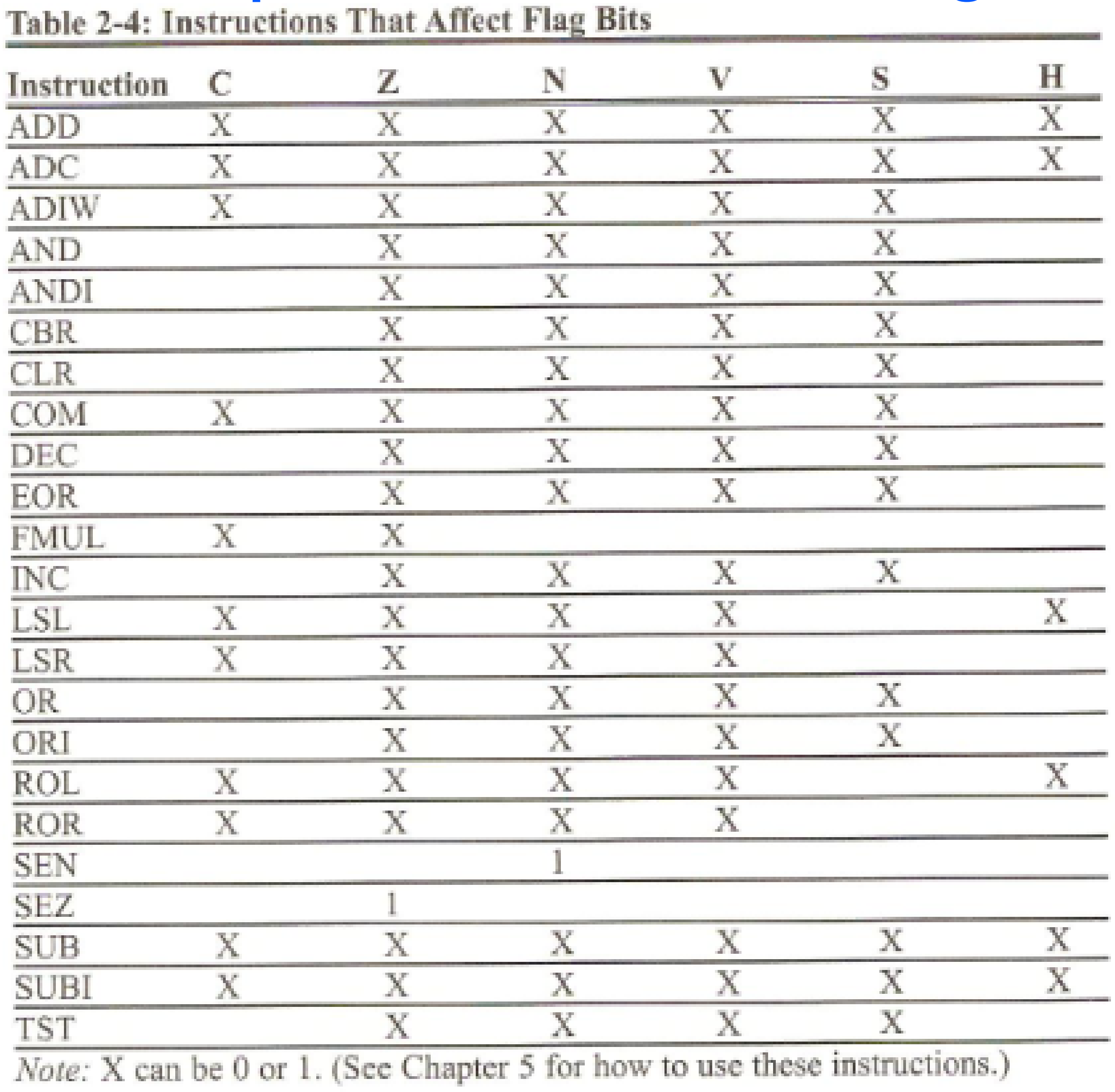
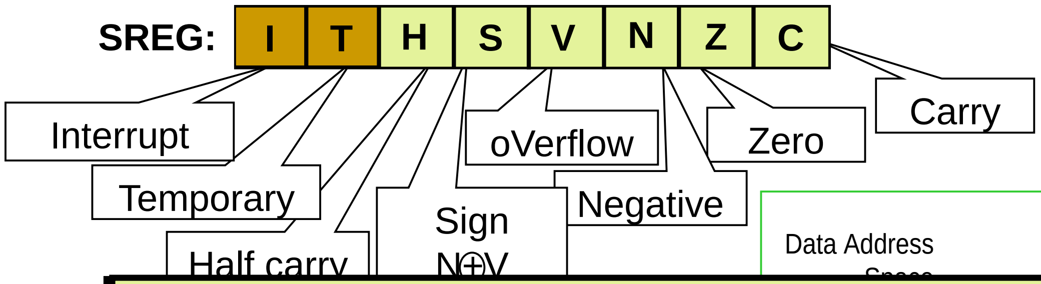


Table 2-4: Instructions That Affect Flag Bits

Instruction	C	Z	N	V	S	H
ADD	X	X	X	X	X	X
ADC	X	X	X	X	X	X
ADIW	X	X	X	X	X	
AND		X	X	X	X	
ANDI		X	X	X	X	
CBR		X	X	X	X	
CLR		X	X	X	X	
COM	X	X	X	X	X	
DEC		X	X	X	X	
EOR		X	X	X	X	
FMUL	X	X				
INC		X	X	X	X	
LSL	X	X	X	X		X
LSR	X	X	X	X		
OR		X	X	X	X	
ORI		X	X	X	X	
ROL	X	X	X	X		X
ROR	X	X	X	X		
SEN			1			
SEZ		1				
SUB	X	X	X	X	X	X
SUBI	X	X	X	X	X	X
TST		X	X	X	X	

Note: X can be 0 or 1. (See Chapter 5 for how to use these instructions.)

Status Register (SREG)



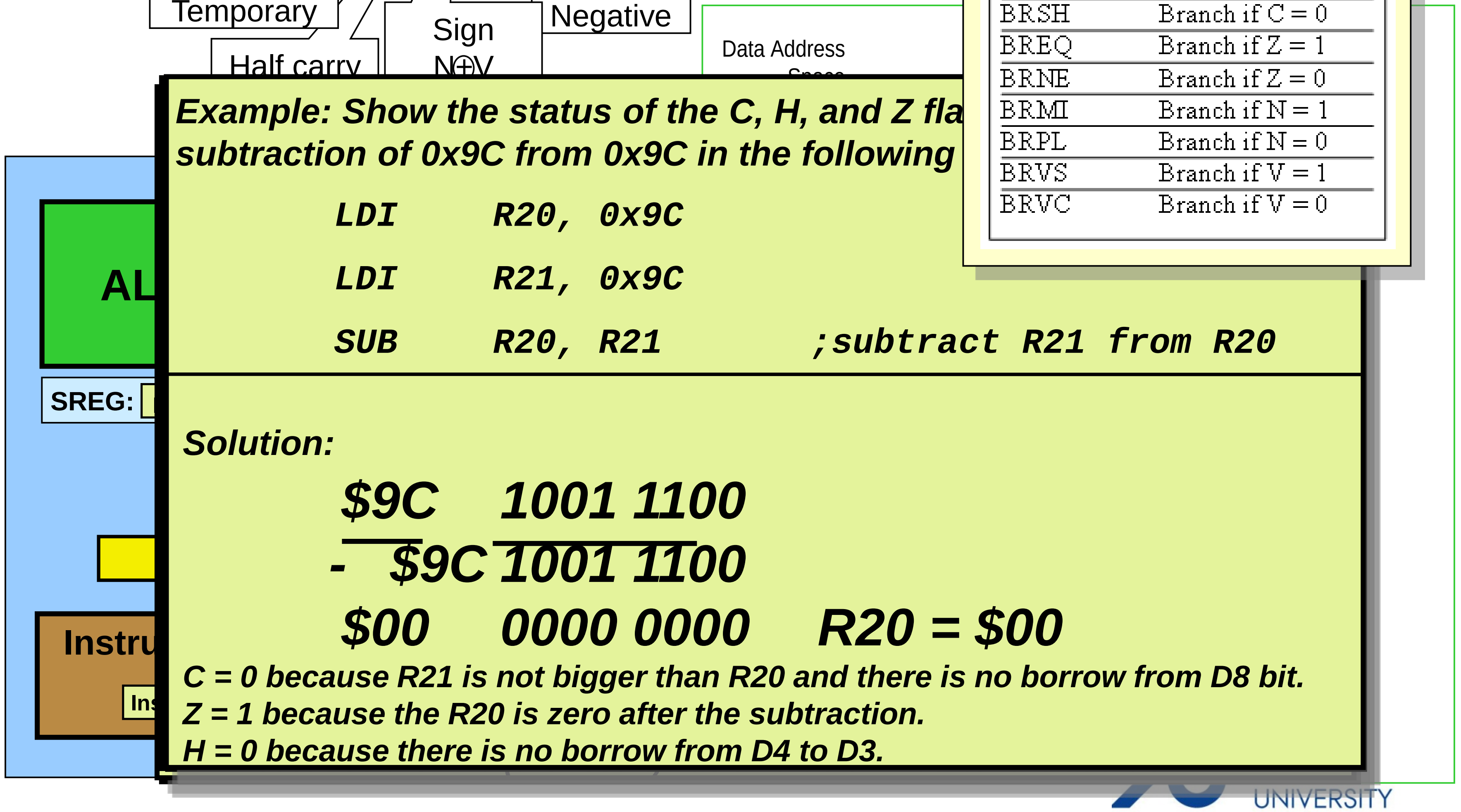


Table 2-5: AVR Branch (Jump) Instructions Using Flag Bits

Instruction	Action
BRLO	Branch if C = 1
BRSH	Branch if C = 0
BREQ	Branch if Z = 1
BRNE	Branch if Z = 0
BRMI	Branch if N = 1
BRPL	Branch if N = 0
BRVS	Branch if V = 1
BRVC	Branch if V = 0

Example: Show the status of the C, H, and Z fla
subtraction of 0x9C from 0x9C in the following

```
        LDI     R20, 0x9C
        LDI     R21, 0x9C
        SUB     R20, R21        ;subtract R21 from R20
```

Solution:

$$\begin{array}{lll} \$9C & 1001\ 1100 & \\ -\ \underline{\$9C} & \underline{1001\ 1100} & \\ \$00 & 0000\ 0000 & R20 = \$00 \end{array}$$

C = 0 because R21 is not bigger than R20 and there is no borrow from D8 bit.
Z = 1 because the R20 is zero after the subtraction.
H = 0 because there is no borrow from D4 to D3.

UNIVERSITY
SCHOOL OF ENGINEERING

Betingede Jumps (bruger flagene)

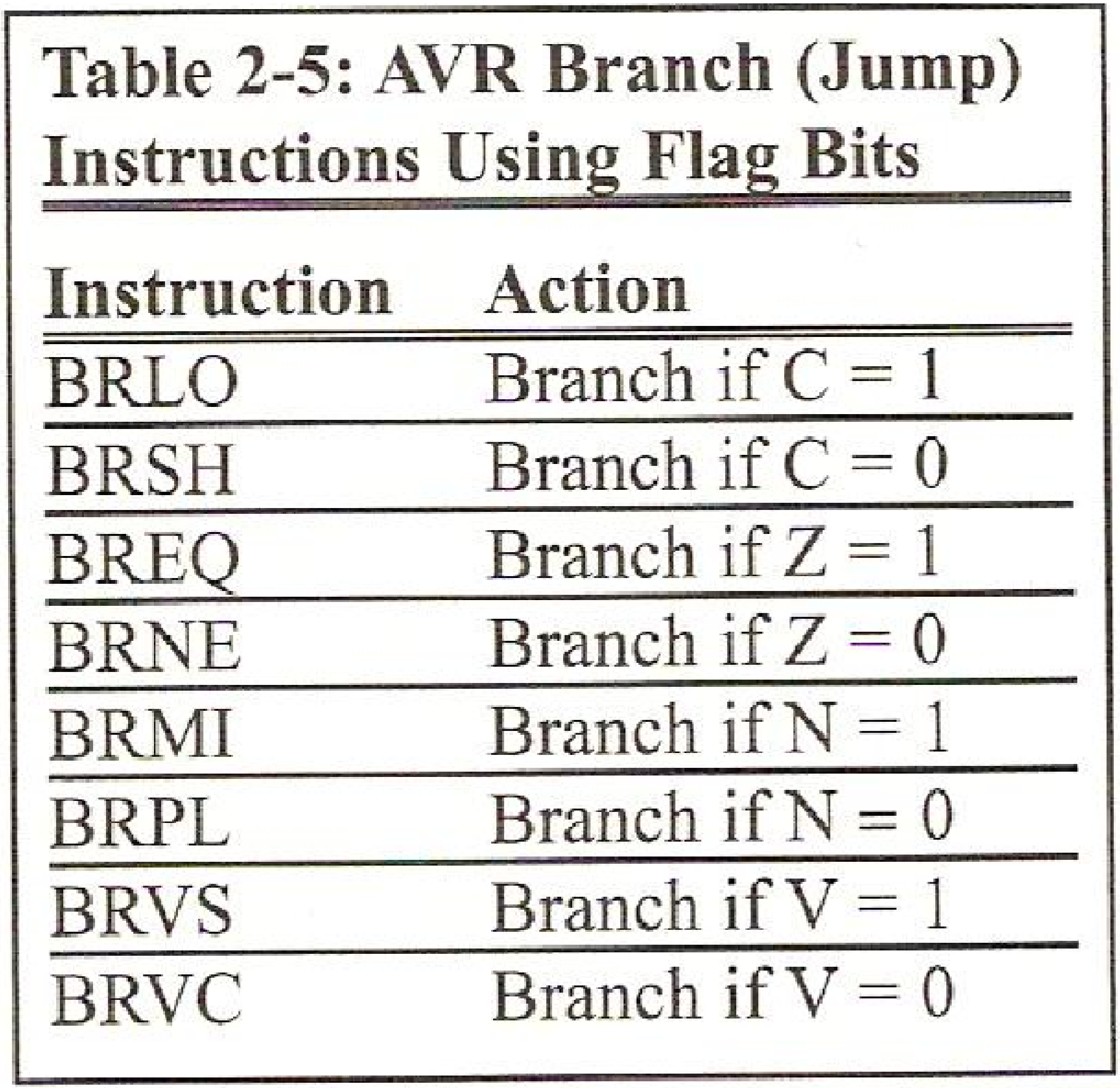
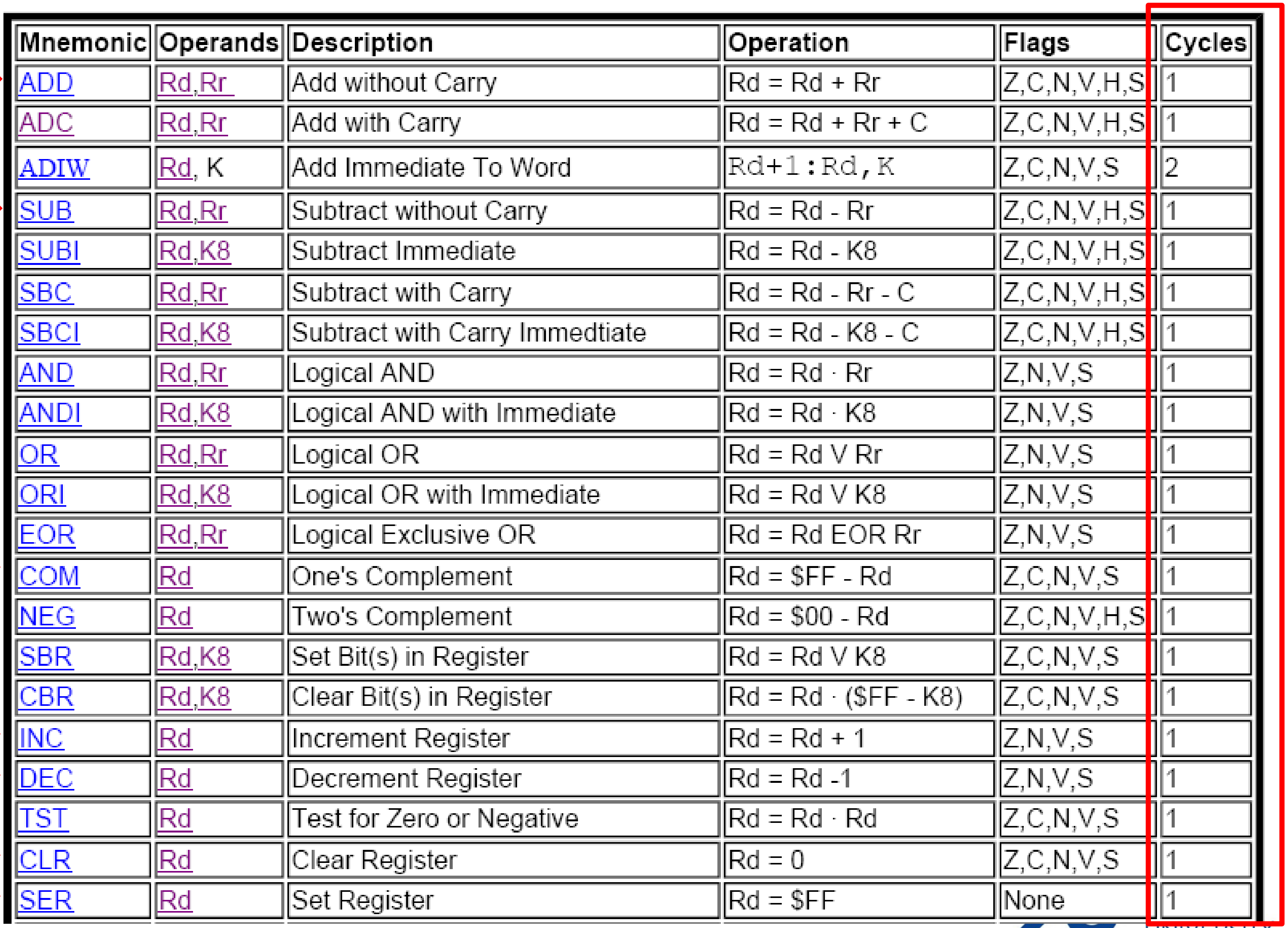


Table 2-5: AVR Branch (Jump) Instructions Using Flag Bits

Instruction	Action
BRLO	Branch if C = 1
BRSH	Branch if C = 0
BREQ	Branch if Z = 1
BRNE	Branch if Z = 0
BRMI	Branch if N = 1
BRPL	Branch if N = 0
BRVS	Branch if V = 1
BRVC	Branch if V = 0

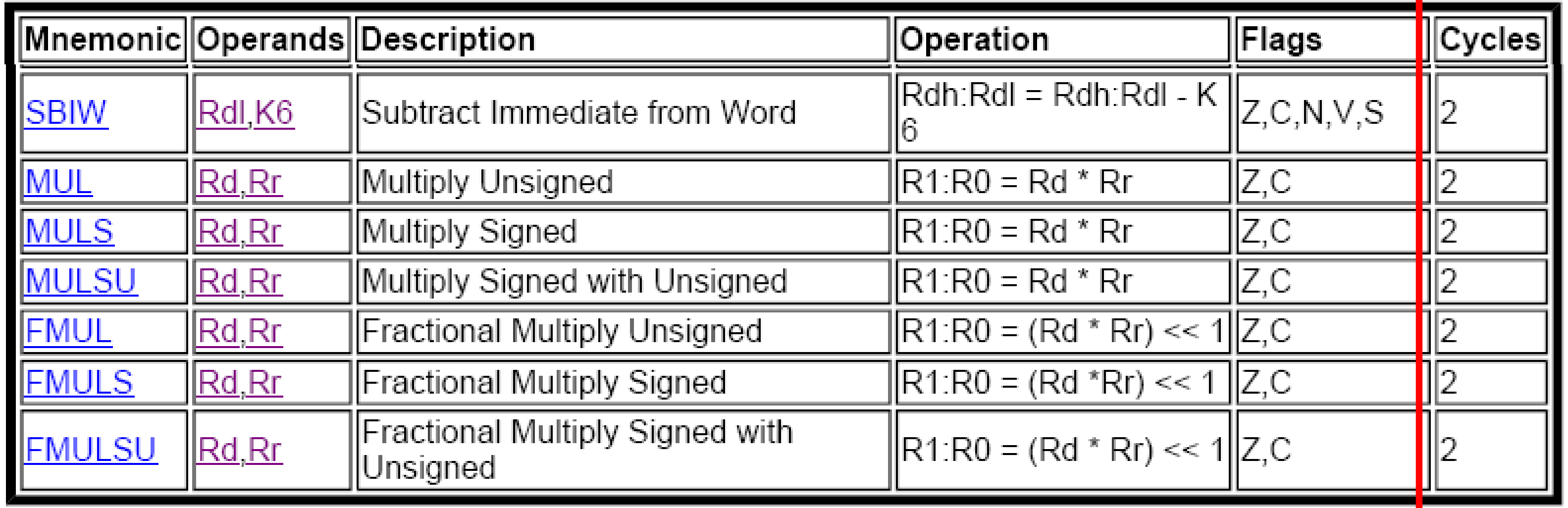
AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Aritmetiske og logiske instruktioner(1)



Mnemonic	Operands	Description	Operation	Flags	Cycles
ADD	Rd,Rr	Add without Carry	Rd = Rd + Rr	Z,C,N,V,H,S	1
ADC	Rd,Rr	Add with Carry	Rd = Rd + Rr + C	Z,C,N,V,H,S	1
ADIW	Rd, K	Add Immediate To Word	Rd+1:Rd,K	Z,C,N,V,S	2
SUB	Rd,Rr	Subtract without Carry	Rd = Rd - Rr	Z,C,N,V,H,S	1
SUBI	Rd,K8	Subtract Immediate	Rd = Rd - K8	Z,C,N,V,H,S	1
SBC	Rd,Rr	Subtract with Carry	Rd = Rd - Rr - C	Z,C,N,V,H,S	1
SBCI	Rd,K8	Subtract with Carry Immedtiate	Rd = Rd - K8 - C	Z,C,N,V,H,S	1
AND	Rd,Rr	Logical AND	Rd = Rd · Rr	Z,N,V,S	1
ANDI	Rd,K8	Logical AND with Immediate	Rd = Rd · K8	Z,N,V,S	1
OR	Rd,Rr	Logical OR	Rd = Rd V Rr	Z,N,V,S	1
ORI	Rd,K8	Logical OR with Immediate	Rd = Rd V K8	Z,N,V,S	1
EOR	Rd,Rr	Logical Exclusive OR	Rd = Rd EOR Rr	Z,N,V,S	1
COM	Rd	One's Complement	Rd = $FF - Rd	Z,C,N,V,S	1
NEG	Rd	Two's Complement	Rd = $00 - Rd	Z,C,N,V,H,S	1
SBR	Rd,K8	Set Bit(s) in Register	Rd = Rd V K8	Z,C,N,V,S	1
CBR	Rd,K8	Clear Bit(s) in Register	Rd = Rd · ($FF - K8)	Z,C,N,V,S	1
INC	Rd	Increment Register	Rd = Rd + 1	Z,N,V,S	1
DEC	Rd	Decrement Register	Rd = Rd -1	Z,N,V,S	1
TST	Rd	Test for Zero or Negative	Rd = Rd · Rd	Z,C,N,V,S	1
CLR	Rd	Clear Register	Rd = 0	Z,C,N,V,S	1
SER	Rd	Set Register	Rd = $FF	None	1

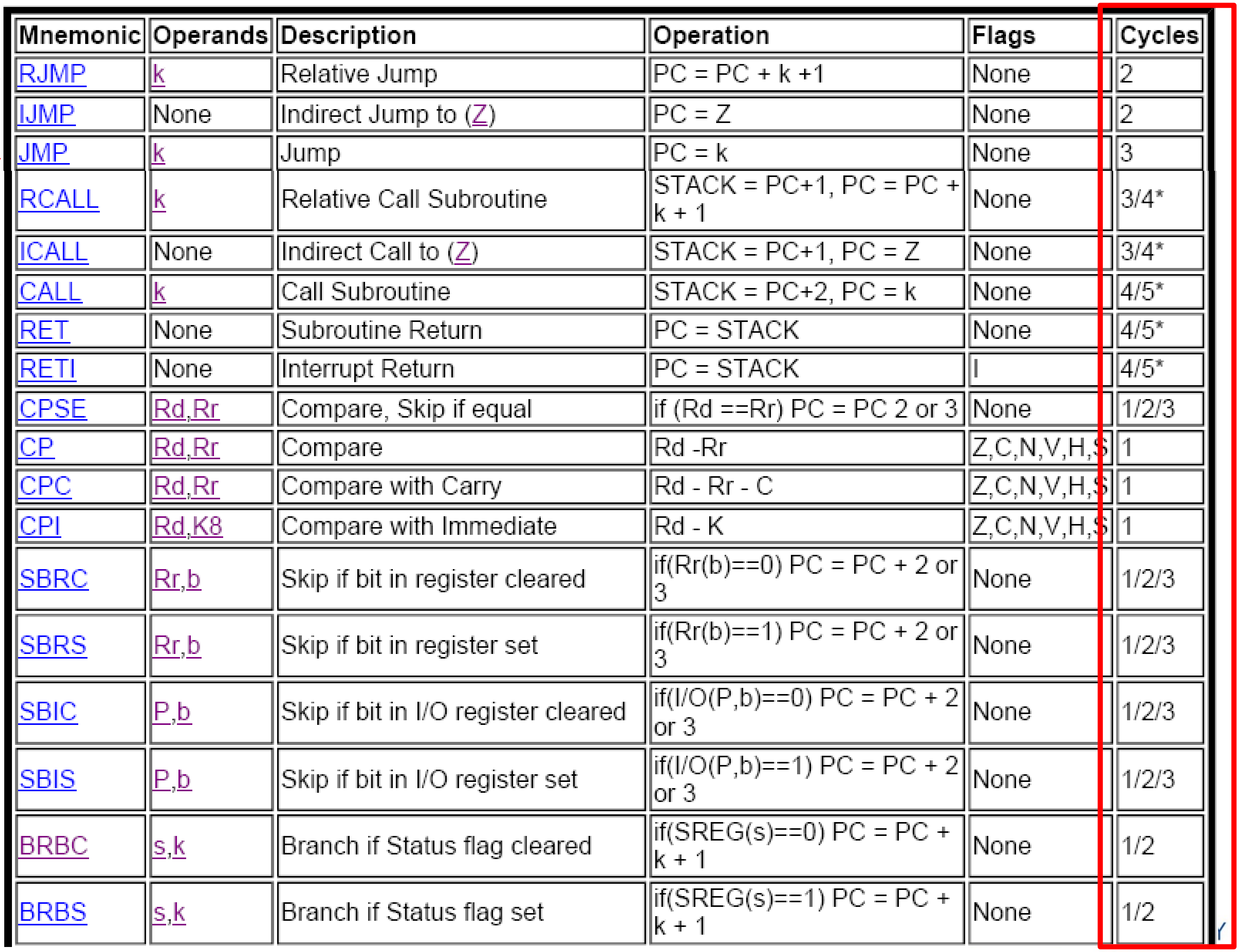
Aritmetiske og logiske instruktioner(2)



Mnemonic	Operands	Description	Operation	Flags	Cycles
SBIW	Rdl,K6	Subtract Immediate from Word	Rdh:Rdl = Rdh:Rdl - K 6	Z,C,N,V,S	2
MUL	Rd,Rr	Multiply Unsigned	R1:R0 = Rd * Rr	Z,C	2
MULS	Rd,Rr	Multiply Signed	R1:R0 = Rd * Rr	Z,C	2
MULSU	Rd,Rr	Multiply Signed with Unsigned	R1:R0 = Rd * Rr	Z,C	2
FMUL	Rd,Rr	Fractional Multiply Unsigned	R1:R0 = (Rd * Rr) << 1	Z,C	2
FMULS	Rd,Rr	Fractional Multiply Signed	R1:R0 = (Rd *Rr) << 1	Z,C	2
FMULSU	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0 = (Rd * Rr) << 1	Z,C	2

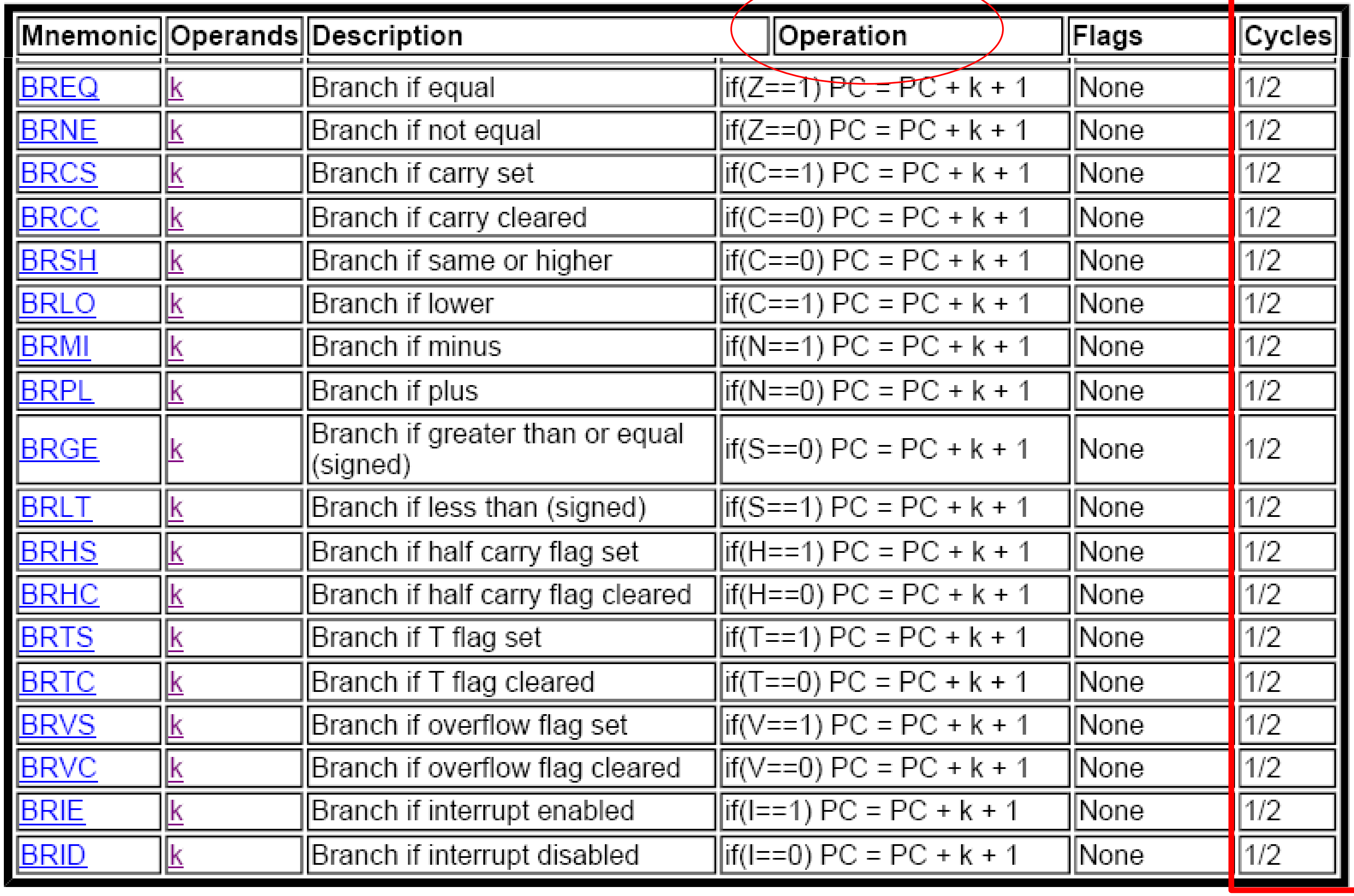
AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Branch instruktioner (1)



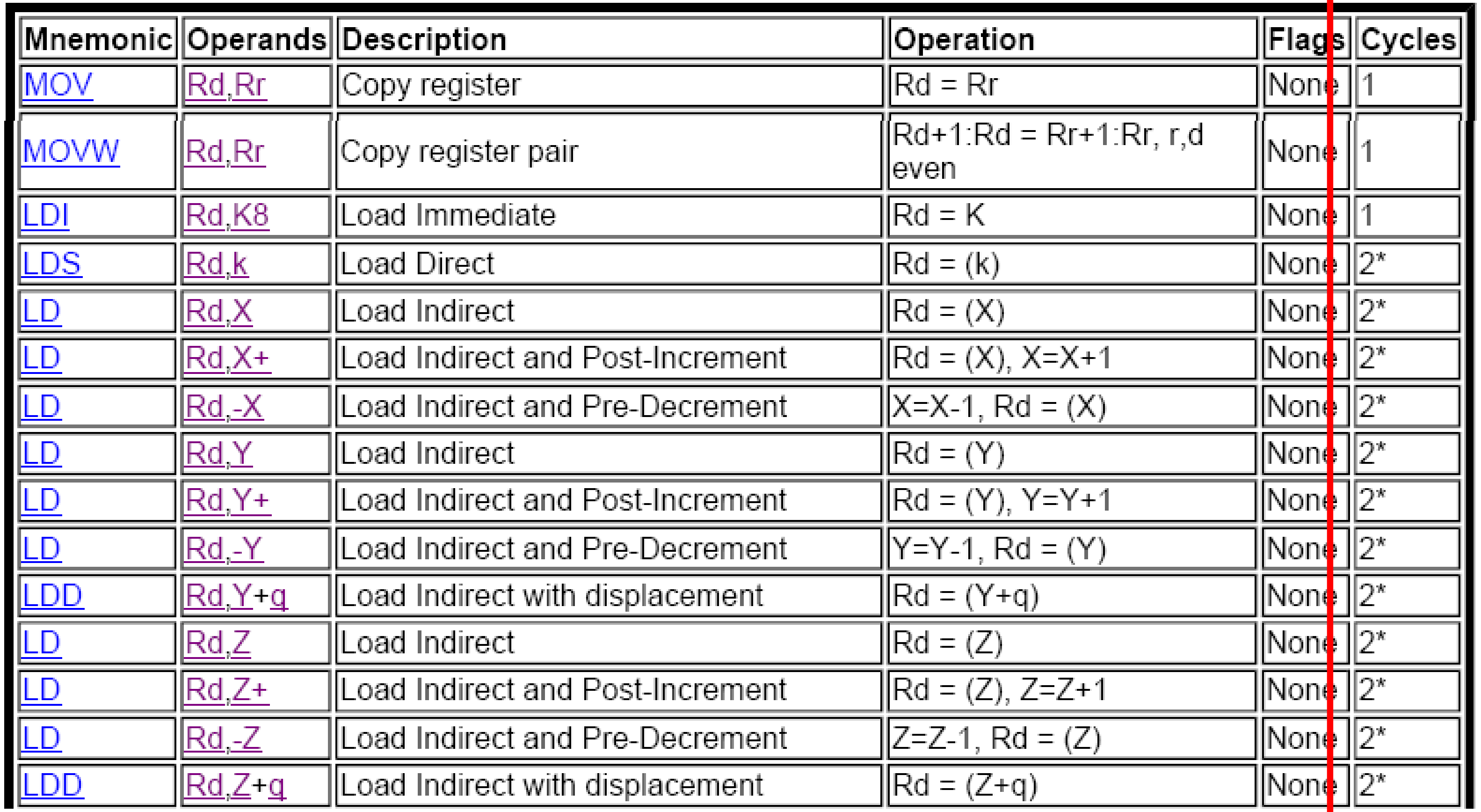
Mnemonic	Operands	Description	Operation	Flags	Cycles
RJMP	k	Relative Jump	PC = PC + k +1	None	2
IJMP	None	Indirect Jump to (Z)	PC = Z	None	2
JMP	k	Jump	PC = k	None	3
RCALL	k	Relative Call Subroutine	STACK = PC+1, PC = PC + k + 1	None	3/4*
ICALL	None	Indirect Call to (Z)	STACK = PC+1, PC = Z	None	3/4*
CALL	k	Call Subroutine	STACK = PC+2, PC = k	None	4/5*
RET	None	Subroutine Return	PC = STACK	None	4/5*
RETI	None	Interrupt Return	PC = STACK	I	4/5*
CPSE	Rd,Rr	Compare, Skip if equal	if (Rd ==Rr) PC = PC 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd -Rr	Z,C,N,V,H,S	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,H,S	1
CPI	Rd,K8	Compare with Immediate	Rd - K	Z,C,N,V,H,S	1
SBRC	Rr,b	Skip if bit in register cleared	if(Rr(b)==0) PC = PC + 2 or 3	None	1/2/3
SBRS	Rr,b	Skip if bit in register set	if(Rr(b)==1) PC = PC + 2 or 3	None	1/2/3
SBIC	P,b	Skip if bit in I/O register cleared	if(I/O(P,b)==0) PC = PC + 2 or 3	None	1/2/3
SBIS	P,b	Skip if bit in I/O register set	if(I/O(P,b)==1) PC = PC + 2 or 3	None	1/2/3
BRBC	s,k	Branch if Status flag cleared	if(SREG(s)==0) PC = PC + k + 1	None	1/2
BRBS	s,k	Branch if Status flag set	if(SREG(s)==1) PC = PC + k + 1	None	1/2

Branch instruktioner (2)



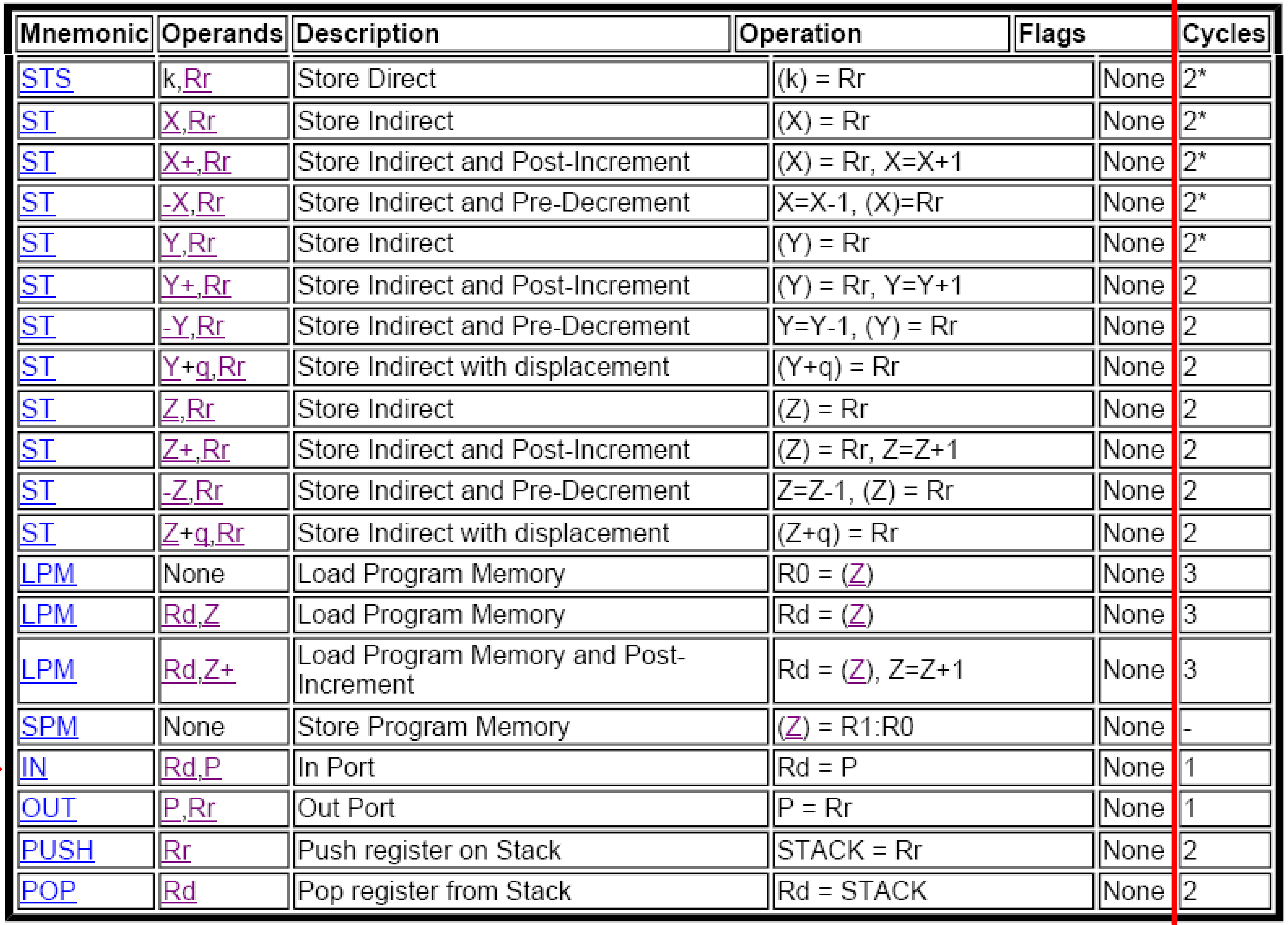
Mnemonic	Operands	Description	Operation	Flags	Cycles
BREQ	k	Branch if equal	if(Z==1) PC = PC + k + 1	None	1/2
BRNE	k	Branch if not equal	if(Z==0) PC = PC + k + 1	None	1/2
BRCS	k	Branch if carry set	if(C==1) PC = PC + k + 1	None	1/2
BRCC	k	Branch if carry cleared	if(C==0) PC = PC + k + 1	None	1/2
BRSH	k	Branch if same or higher	if(C==0) PC = PC + k + 1	None	1/2
BRLO	k	Branch if lower	if(C==1) PC = PC + k + 1	None	1/2
BRMI	k	Branch if minus	if(N==1) PC = PC + k + 1	None	1/2
BRPL	k	Branch if plus	if(N==0) PC = PC + k + 1	None	1/2
BRGE	k	Branch if greater than or equal (signed)	if(S==0) PC = PC + k + 1	None	1/2
BRLT	k	Branch if less than (signed)	if(S==1) PC = PC + k + 1	None	1/2
BRHS	k	Branch if half carry flag set	if(H==1) PC = PC + k + 1	None	1/2
BRHC	k	Branch if half carry flag cleared	if(H==0) PC = PC + k + 1	None	1/2
BRTS	k	Branch if T flag set	if(T==1) PC = PC + k + 1	None	1/2
BRTC	k	Branch if T flag cleared	if(T==0) PC = PC + k + 1	None	1/2
BRVS	k	Branch if overflow flag set	if(V==1) PC = PC + k + 1	None	1/2
BRVC	k	Branch if overflow flag cleared	if(V==0) PC = PC + k + 1	None	1/2
BRIE	k	Branch if interrupt enabled	if(I==1) PC = PC + k + 1	None	1/2
BRID	k	Branch if interrupt disabled	if(I==0) PC = PC + k + 1	None	1/2

Data transfer instruktioner (1)



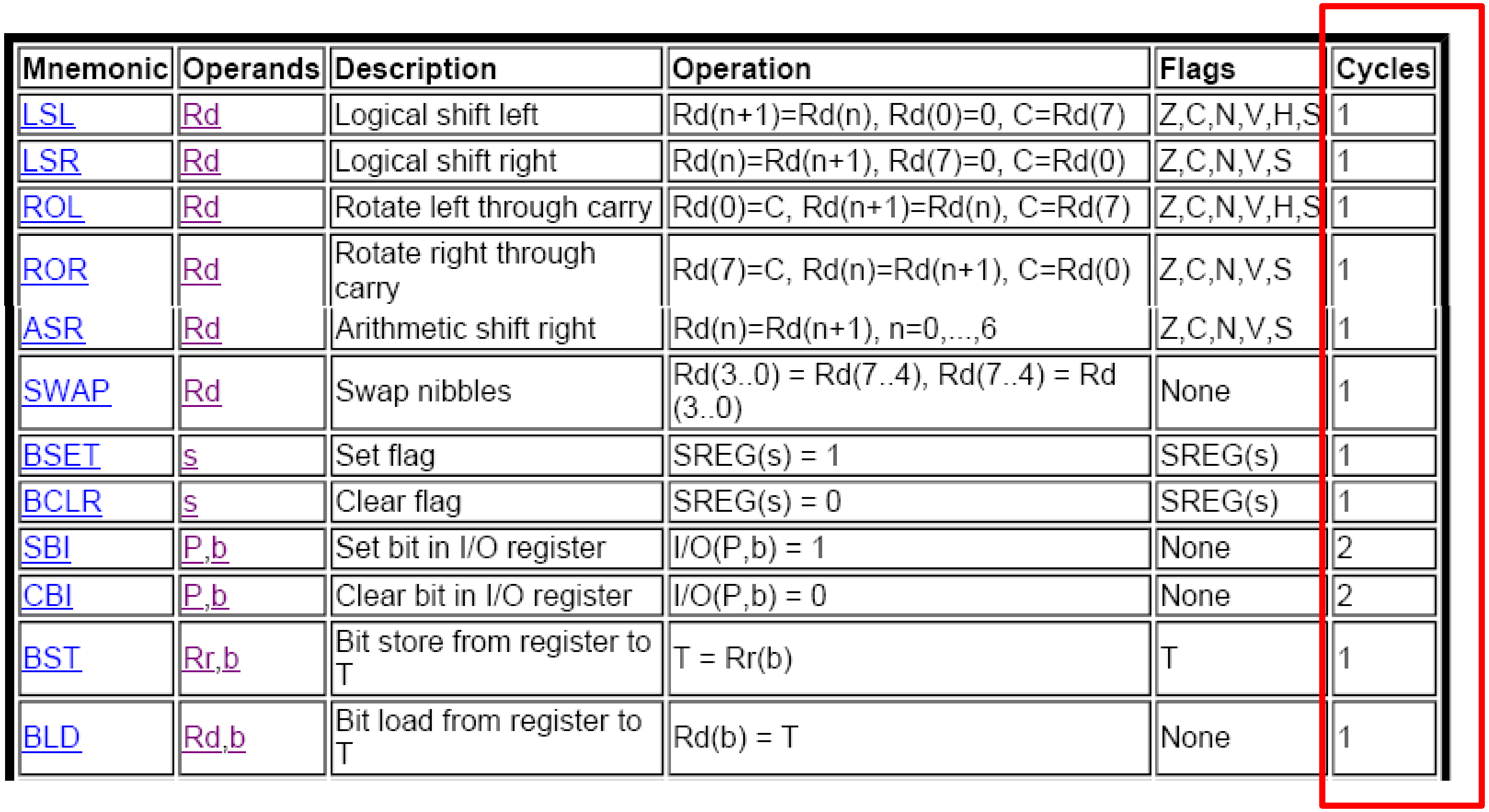
Mnemonic	Operands	Description	Operation	Flags	Cycles
MOV	Rd,Rr	Copy register	Rd = Rr	None	1
MOVW	Rd,Rr	Copy register pair	Rd+1:Rd = Rr+1:Rr, r,d even	None	1
LDI	Rd,K8	Load Immediate	Rd = K	None	1
LDS	Rd,k	Load Direct	Rd = (k)	None	2*
LD	Rd,X	Load Indirect	Rd = (X)	None	2*
LD	Rd,X+	Load Indirect and Post-Increment	Rd = (X), X=X+1	None	2*
LD	Rd,-X	Load Indirect and Pre-Decrement	X=X-1, Rd = (X)	None	2*
LD	Rd,Y	Load Indirect	Rd = (Y)	None	2*
LD	Rd,Y+	Load Indirect and Post-Increment	Rd = (Y), Y=Y+1	None	2*
LD	Rd,-Y	Load Indirect and Pre-Decrement	Y=Y-1, Rd = (Y)	None	2*
LDD	Rd,Y+q	Load Indirect with displacement	Rd = (Y+q)	None	2*
LD	Rd,Z	Load Indirect	Rd = (Z)	None	2*
LD	Rd,Z+	Load Indirect and Post-Increment	Rd = (Z), Z=Z+1	None	2*
LD	Rd,-Z	Load Indirect and Pre-Decrement	Z=Z-1, Rd = (Z)	None	2*
LDD	Rd,Z+q	Load Indirect with displacement	Rd = (Z+q)	None	2*

Data transfer instruktioner (2)



Mnemonic	Operands	Description	Operation	Flags	Cycles
STS	k,Rr	Store Direct	(k) = Rr	None	2*
ST	X,Rr	Store Indirect	(X) = Rr	None	2*
ST	X+,Rr	Store Indirect and Post-Increment	(X) = Rr, X=X+1	None	2*
ST	-X,Rr	Store Indirect and Pre-Decrement	X=X-1, (X)=Rr	None	2*
ST	Y,Rr	Store Indirect	(Y) = Rr	None	2*
ST	Y+,Rr	Store Indirect and Post-Increment	(Y) = Rr, Y=Y+1	None	2
ST	-Y,Rr	Store Indirect and Pre-Decrement	Y=Y-1, (Y) = Rr	None	2
ST	Y+q,Rr	Store Indirect with displacement	(Y+q) = Rr	None	2
ST	Z,Rr	Store Indirect	(Z) = Rr	None	2
ST	Z+,Rr	Store Indirect and Post-Increment	(Z) = Rr, Z=Z+1	None	2
ST	-Z,Rr	Store Indirect and Pre-Decrement	Z=Z-1, (Z) = Rr	None	2
ST	Z+q,Rr	Store Indirect with displacement	(Z+q) = Rr	None	2
LPM	None	Load Program Memory	R0 = (Z)	None	3
LPM	Rd,Z	Load Program Memory	Rd = (Z)	None	3
LPM	Rd,Z+	Load Program Memory and Post-Increment	Rd = (Z), Z=Z+1	None	3
SPM	None	Store Program Memory	(Z) = R1:R0	None	-
IN	Rd,P	In Port	Rd = P	None	1
OUT	P,Rr	Out Port	P = Rr	None	1
PUSH	Rr	Push register on Stack	STACK = Rr	None	2
POP	Rd	Pop register from Stack	Rd = STACK	None	2

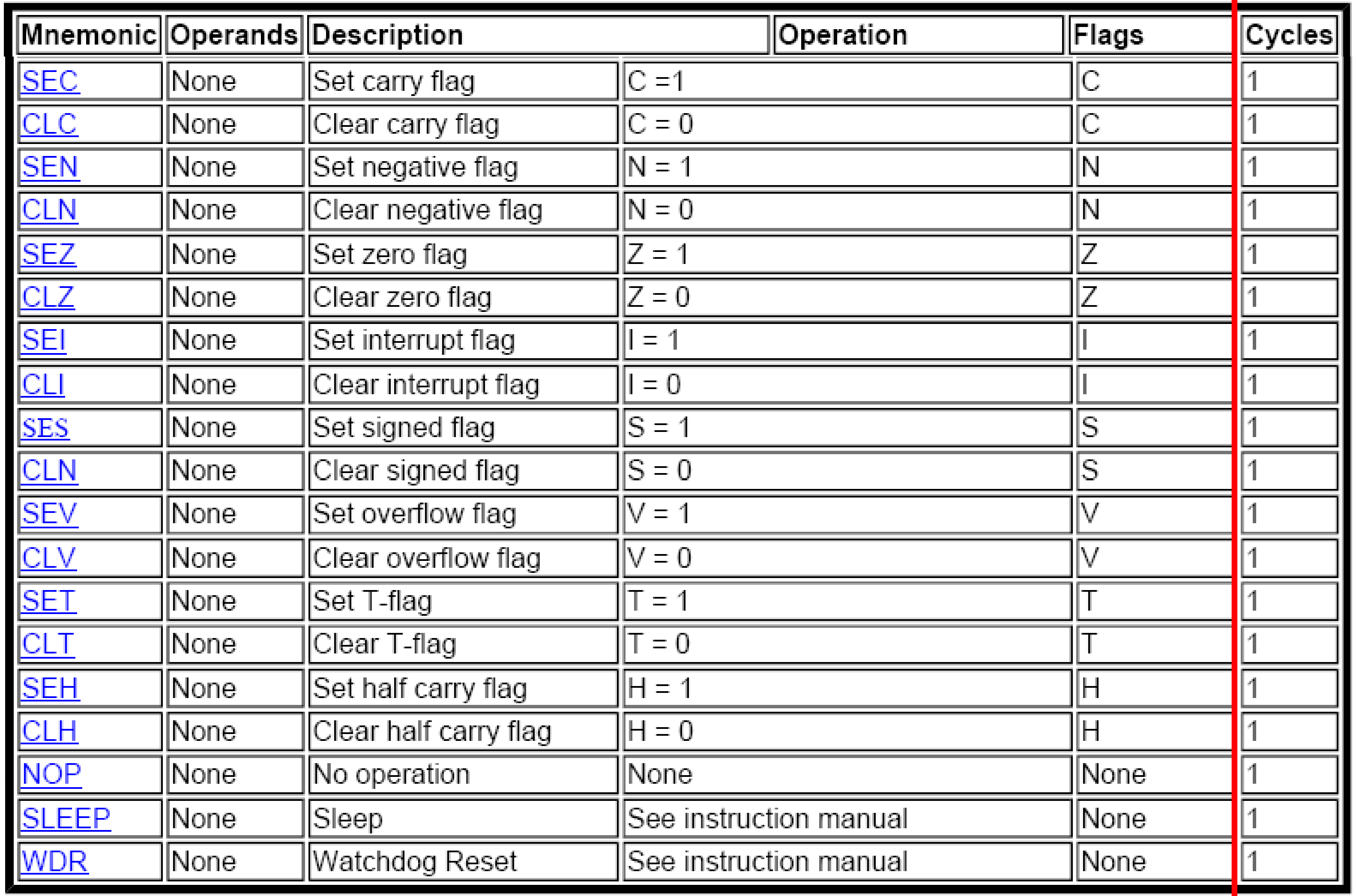
Bit- og Bit Test instruktioner (1)



Mnemonic	Operands	Description	Operation	Flags	Cycles
LSL	Rd	Logical shift left	Rd(n+1)=Rd(n), Rd(0)=0, C=Rd(7)	Z,C,N,V,H,S	1
LSR	Rd	Logical shift right	Rd(n)=Rd(n+1), Rd(7)=0, C=Rd(0)	Z,C,N,V,S	1
ROL	Rd	Rotate left through carry	Rd(0)=C, Rd(n+1)=Rd(n), C=Rd(7)	Z,C,N,V,H,S	1
ROR	Rd	Rotate right through carry	Rd(7)=C, Rd(n)=Rd(n+1), C=Rd(0)	Z,C,N,V,S	1
ASR	Rd	Arithmetic shift right	Rd(n)=Rd(n+1), n=0,...,6	Z,C,N,V,S	1
SWAP	Rd	Swap nibbles	Rd(3..0) = Rd(7..4), Rd(7..4) = Rd (3..0)	None	1
BSET	s	Set flag	SREG(s) = 1	SREG(s)	1
BCLR	s	Clear flag	SREG(s) = 0	SREG(s)	1
SBI	P,b	Set bit in I/O register	I/O(P,b) = 1	None	2
CBI	P,b	Clear bit in I/O register	I/O(P,b) = 0	None	2
BST	Rr,b	Bit store from register to T	T = Rr(b)	T	1
BLD	Rd,b	Bit load from register to T	Rd(b) = T	None	1

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Bit- og Bit Test instruktioner (2)



Mnemonic	Operands	Description	Operation	Flags	Cycles
SEC	None	Set carry flag	C =1	C	1
CLC	None	Clear carry flag	C = 0	C	1
SEN	None	Set negative flag	N = 1	N	1
CLN	None	Clear negative flag	N = 0	N	1
SEZ	None	Set zero flag	Z = 1	Z	1
CLZ	None	Clear zero flag	Z = 0	Z	1
SEI	None	Set interrupt flag	I = 1	I	1
CLI	None	Clear interrupt flag	I = 0	I	1
SES	None	Set signed flag	S = 1	S	1
CLN	None	Clear signed flag	S = 0	S	1
SEV	None	Set overflow flag	V = 1	V	1
CLV	None	Clear overflow flag	V = 0	V	1
SET	None	Set T-flag	T = 1	T	1
CLT	None	Clear T-flag	T = 0	T	1
SEH	None	Set half carry flag	H = 1	H	1
CLH	None	Clear half carry flag	H = 0	H	1
NOP	None	No operation	None	None	1
SLEEP	None	Sleep	See instruction manual	None	1
WDR	None	Watchdog Reset	See instruction manual	None	1

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

NOP (No Operation)

Description:

This instruction performs a single cycle No Operation.

	Operation:
(i)	No

	Syntax:	Operands:	Program Counter:
(i)	NOP	None	PC ← PC + 1

16-bit Opcode:

0000	0000	0000	0000

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
        CLR  R16          ; Clear r16
        SER  R17          ; Set r17
        OUT  PORTB,R16    ; Write zeros to Port B
        NOP               ; Wait (do nothing)
        OUT  PORTB,R17    ; Write ones to Port B
```

Words: 1 (2 bytes)

Cycles: 1

Time delay: Eksempel

```
                              machine cycle
LDI     R16, 19                     1
LDI     R20, 95                     1
LDI     R21, 5                      1
ADD     R16, R20                    1
ADD     R16, R21                    1
                                 ───────
                                    5
```

Time delay: Eksempel

```
                                  machine cycle
        LDI     R16, 100                1
AGAIN:  ADD     R17,R16                 1           *100
        DEC     R16                     1           *100
        BRNE    AGAIN                  1/2          *100
```

Branch penalty

Time delay: Eksempel

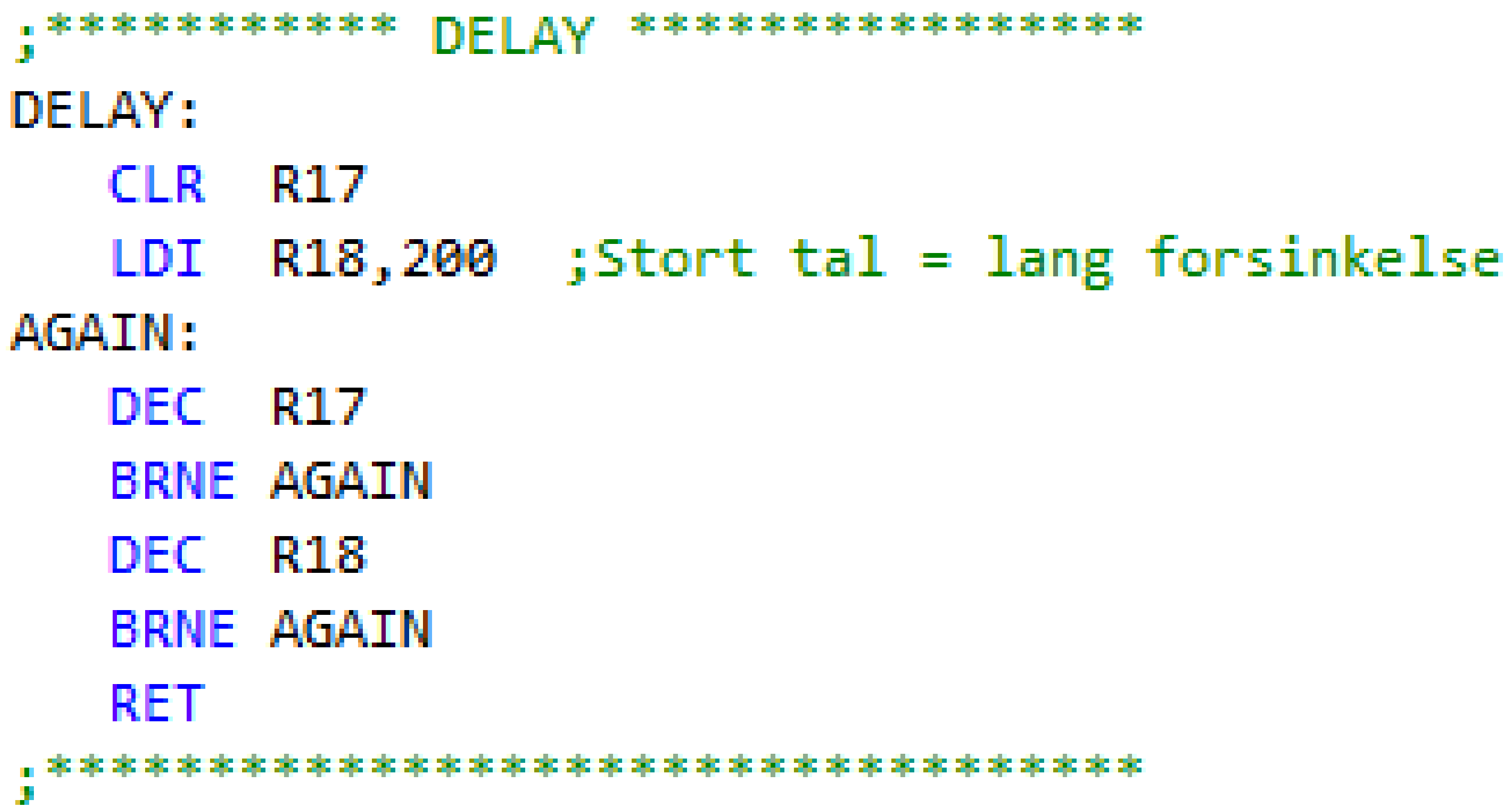
```
                                  machine cycle
        LDI     R16, 50                1
AGAIN:  NOP                            1           *50
        NOP                            1           *50
        DEC     R16                    1           *50
        BRNE    AGAIN                 1/2          *50
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Time delay: Eksempel

```
                                          machine cycle
          LDI     R17, 20                     1
L1:       LDI     R16, 50                     1        *20
L2:       NOP                                 1        *20 * 50
          NOP                                 1        *20 * 50
          DEC     R16                         1        *20 * 50
          BRNE    L2                         1/2       *20 * 50
          DEC     R17                         1        *20
          BRNE    L1                         1/2       *20
                                           ______
```

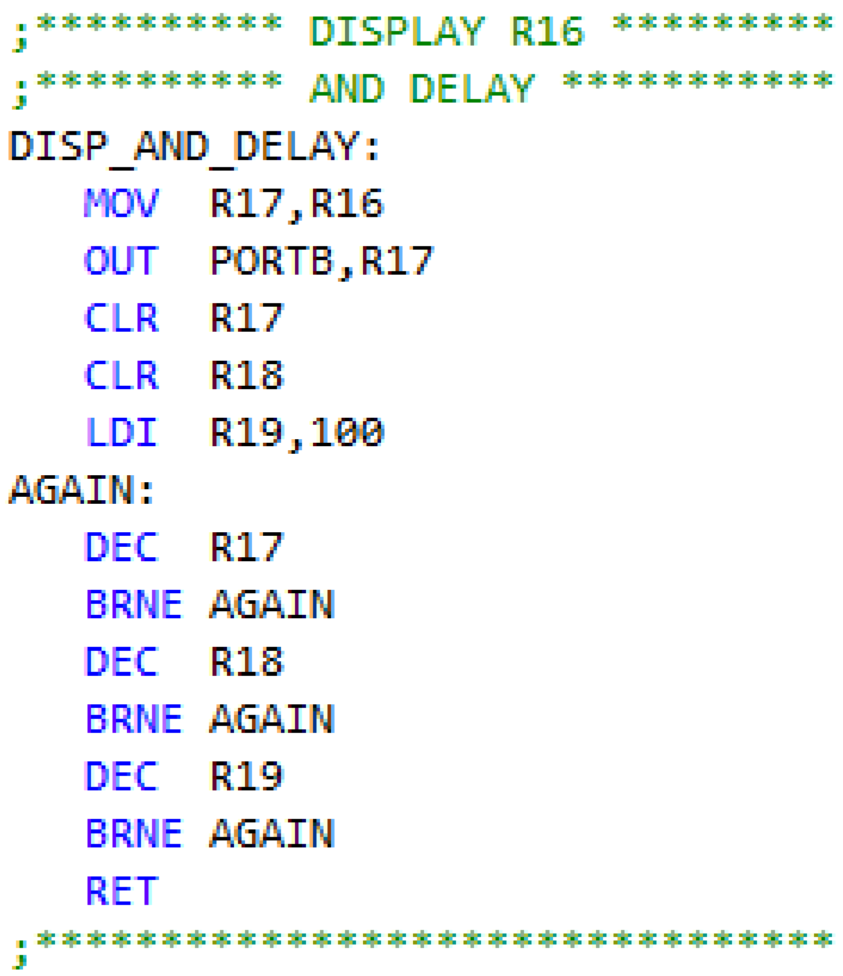
Kode fra LAB2



```
;*********** DELAY ****************
DELAY:
    CLR   R17
    LDI   R18,200   ;Stort tal = lang forsinkelse
AGAIN:
    DEC   R17
    BRNE  AGAIN
    DEC   R18
    BRNE  AGAIN
    RET
;**********************************
```

OPGAVE: Hvor stor er tidsforsinkelsen i DELAY-funktionen?
Svar på ”socrative.com”: Room = MSYS

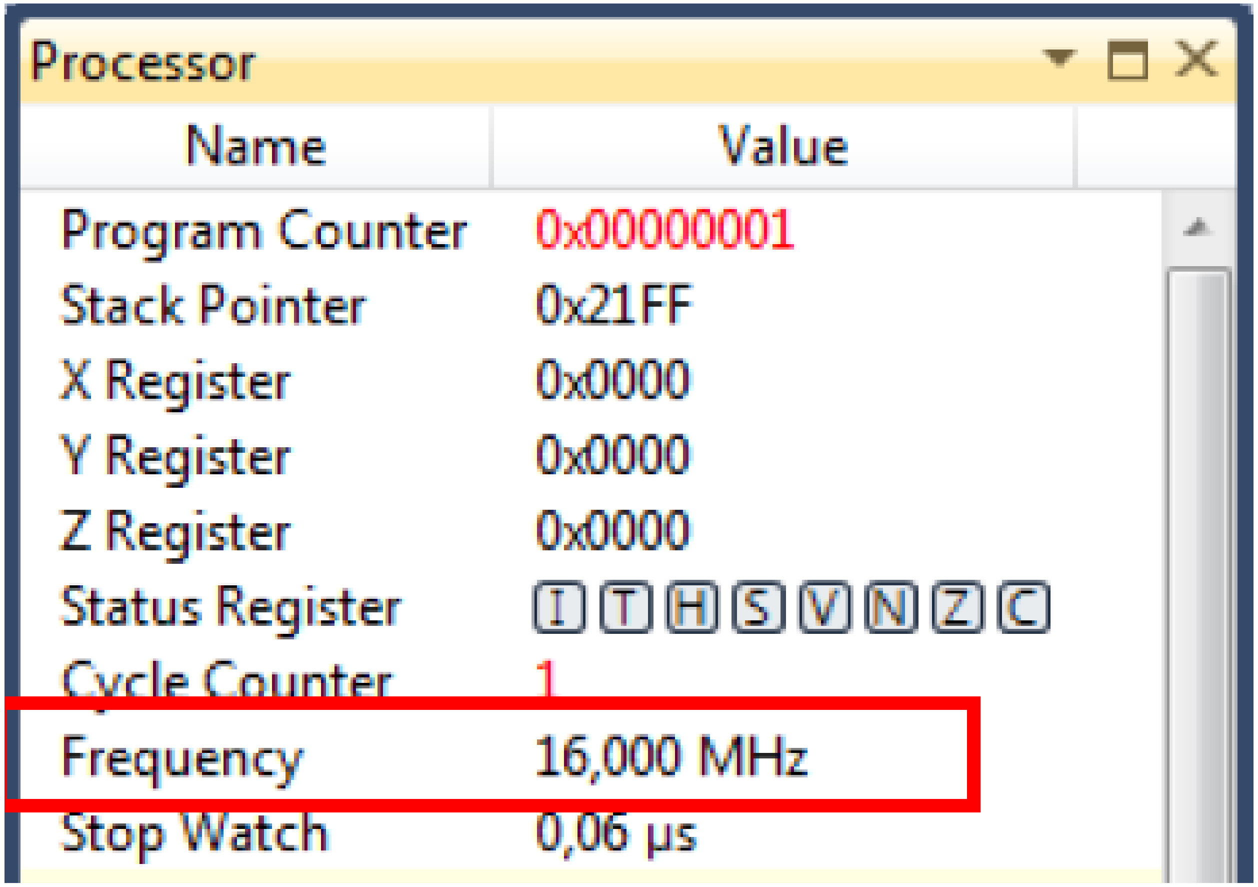
Kode fra LAB1



```
;*********** DISPLAY R16 **********
;*********** AND DELAY ***********
DISP_AND_DELAY:
    MOV   R17,R16
    OUT   PORTB,R17
    CLR   R17
    CLR   R18
    LDI   R19,100
AGAIN:
    DEC   R17
    BRNE  AGAIN
    DEC   R18
    BRNE  AGAIN
    DEC   R19
    BRNE  AGAIN
    RET
;********************************
```

OPGAVE: Hvor stor er tidsforsinkelsen i alt ?
Svar på ”socrative.com”: Room = MSYS

Simulatorens stopur (og cycle counter)

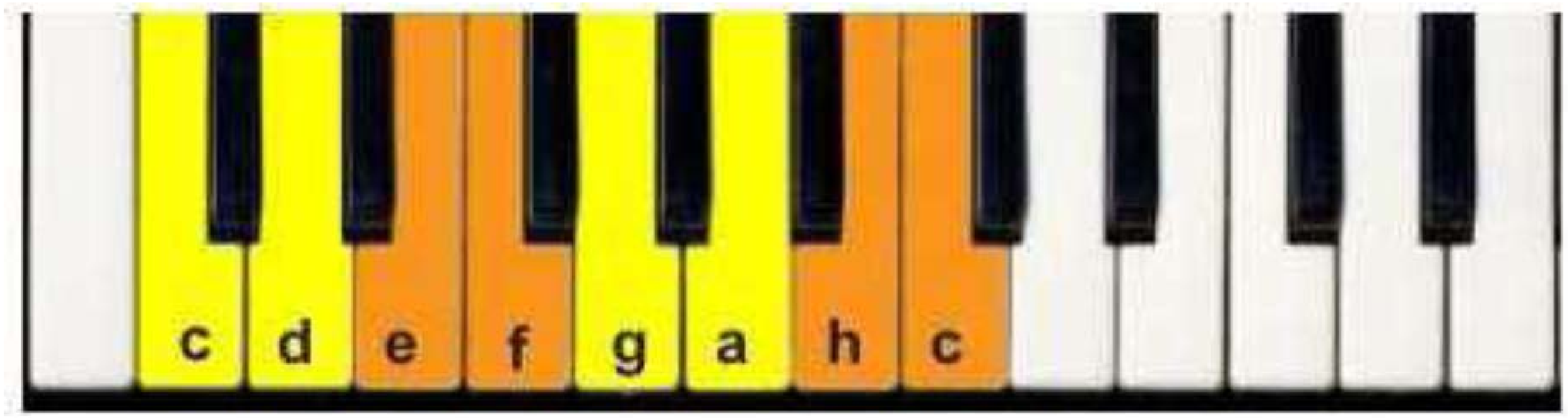


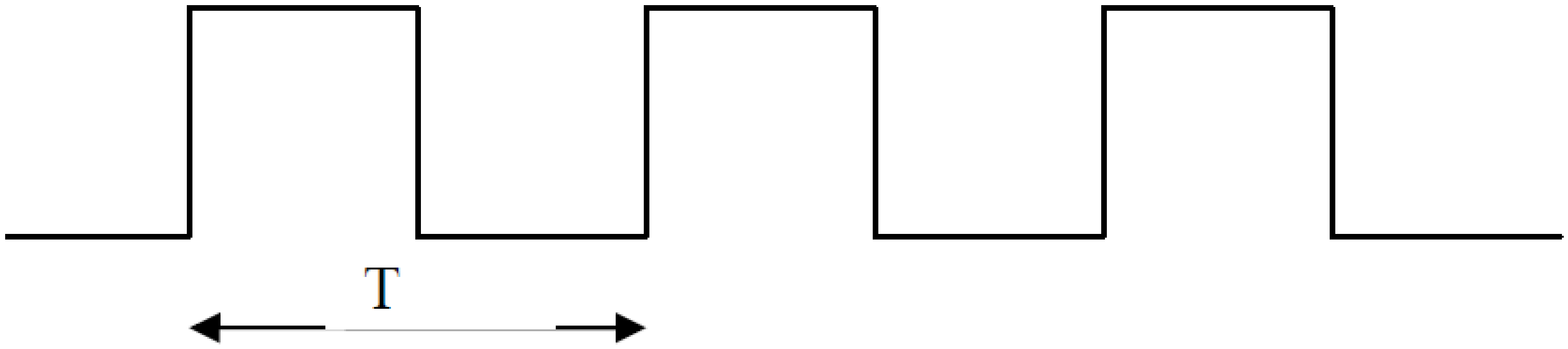
Processor

Name	Value
Program Counter	0x00000001
Stack Pointer	0x21FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	1
Frequency	16,000 MHz
Stop Watch	0,06 µs

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Oplæg til LAB3



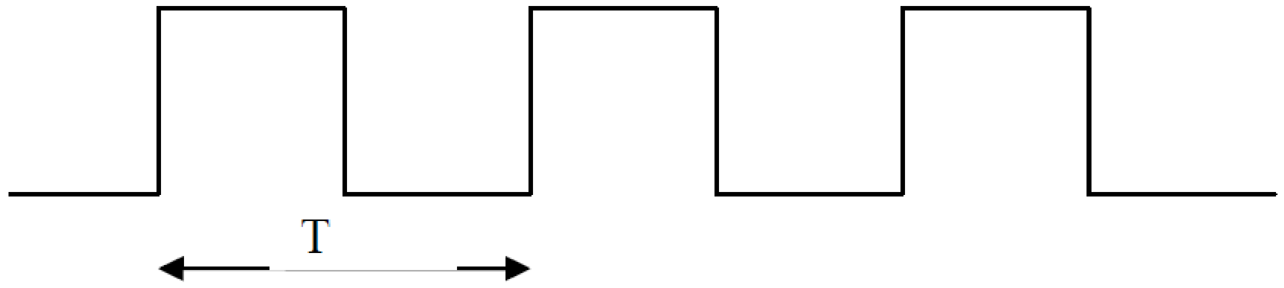


AARHUS UNIVERSITY SCHOOL OF ENGINEERING

Oplæg til LAB3

Tone	Frekvens	T = 1/f	T/2	(T/2)/(4us)
c	523,25 Hz	1911 us	956 us	239
D	587,33 Hz	1792 us	851 us	213
E	659,26 Hz	1517 us	758 us	190
F	698,46 Hz	1432 us	716 us	179
G	783,99 Hz	1276 us	638 us	160
A	880,00 Hz	1136 us	568 us	142
H	987,77 Hz	1012 us	506 us	127
C	1046,50 Hz	956 us	478 us	120

Tabel 1: Frekvenserne + beregninger for en C dur skala.



AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Slut på lektion 6

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING