

# Lektion 3

## Hukommelse

Emil Lykke Diget

Computerarkitektur og Operativsystemer  
Syddansk Universitet

# Introduction

## Computerarkitektur og Operativsystemer

### Viden

- Kombinatoriske logiske kredse
- **Memorytyper**
- **Memoryinterface incl. timing**
- **Adressedekodning**
- Interrupt og exceptions
- Computerdesign
- Register-transfer level
- Datapath
- Control unit
- Instruktionssæt
- Pipeline
- Cache
- Processer og tråde
- Context switch
- Inter-process synkronisering og kommunikation, kritiske sektorer og semaphores

### Færdigheder

- Redegøre for principperne og algoritmerne bag operativsystemets centrale funktioner
- **Forstå opbygningen af en moderne CPU**
- **Kende de almindeligt forekomne memorytyper**
- Forstå centrale begreber omkring et operativsystems afvikling af et program

### Kompetencer

- Implementere operativsystemsfunktioner i et RTOS (Real Time Operating System)

- Lektion 1: Kombinatoriske Logiske Kredse
- Lektion 2: En Simpel Computer
- Lektion 3: Hukommelse
- Lektion 4: Mikroarkitektur
- Lektion 5: Micro-assembly og IJVM
- Lektion 6: Optimering af Mikroarkitekturdesign

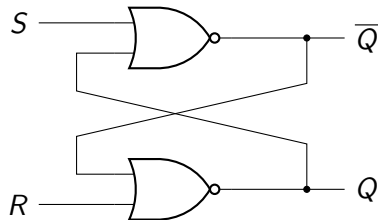
Hukommelse i en computer kan deles op i to kategorier:

- Volatil (RAM)
- Ikke-volatil (ROM)

# Introduktion

## SR Latch

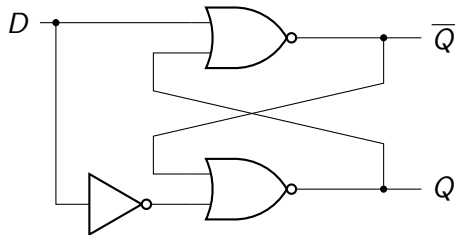
- Værdien af en latch er **ikke** bestemt af de nuværende input.
- To input:
  - ▶  $S$ : Set
  - ▶  $R$ : Reset
  - ▶ Må **ikke** tændes samtidigt.
- To stabile tilstande:
  - ▶  $Q = 1, \overline{Q} = 0$
  - ▶  $Q = 0, \overline{Q} = 1$



# Introduktion

## D Latch

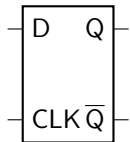
- SR latch er ustabil, når  $S = R = 1$ , så det skal undgås.



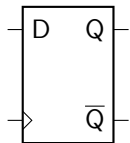
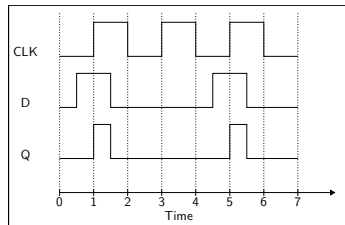
# Introduktion

## Latches og Flip-Flops

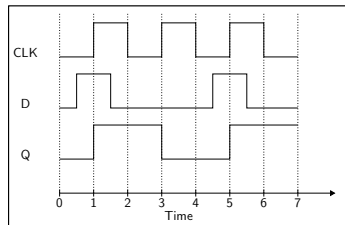
Kan gemme 1 bit!



Figur: D Latch

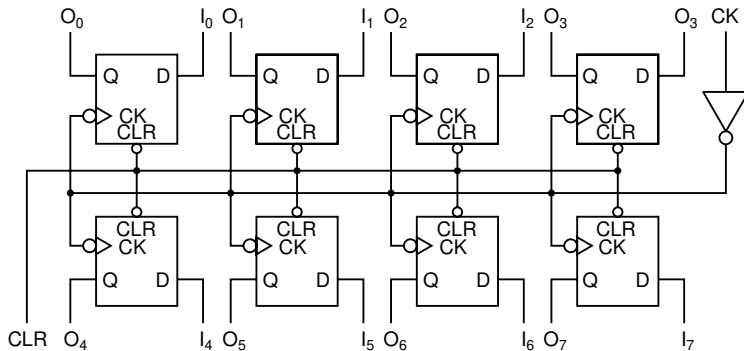


Figur: D Flip-Flop



Otte D flip-flop kan arrangeres for at lave et 8-bit register.

- CLR: Clear
- CK: Clock





## Hukommelsesinterface

- 8-bit Interface

- 16-bit Interface

## Wait State

- Intel 8088

- ATmega128L

## Adresse-demultiplexer

## ATmega128L

## Opsummering

## Referencer

## Hukommelsesinterface

- 8-bit Interface

- 16-bit Interface

Wait State

Adresse-demultiplexer

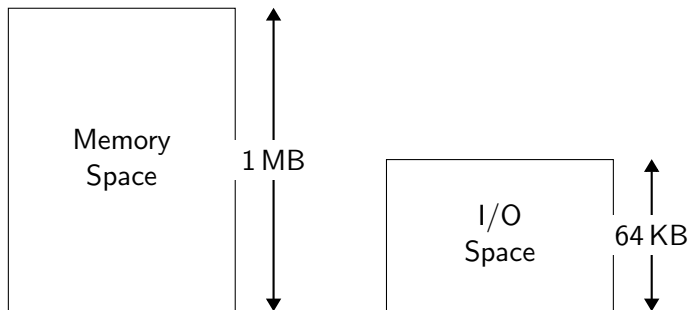
ATmega128L

Opsummering

Referencer

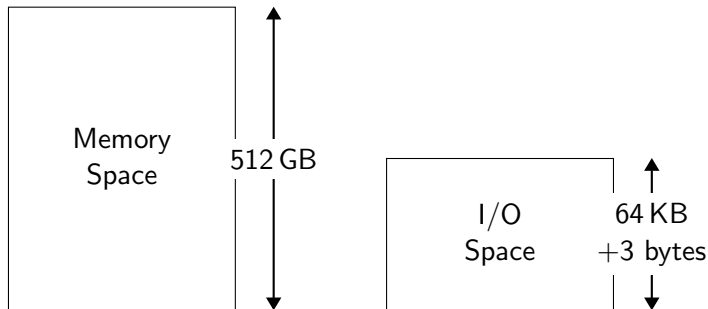
# Hukommelsesinterface (Memory Interface)

Hukommelse- og I/O-område, f.eks. Intel 8088 [1].



# Hukommelsesinterface

Intel 12. generation

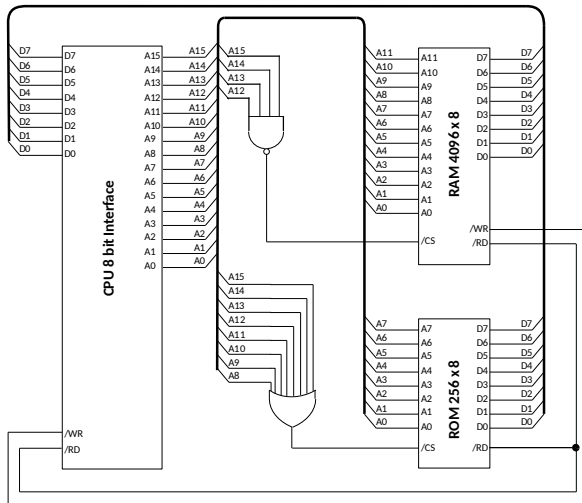


I denne lektion vil vi kigge på hukommelsen og hvordan det tilgås, altså hukommelsesinterface.

I/O indeholder meget forskelligt, f.eks. serial, netværkskort, skærme, etc. Vi skipper det.

# Hukommelsesinterface: 8-bit

8-bit interface med hardwired adresse-dekodning



# Hukommelsesinterface: 8-bit

## Memory map

RAM (random-access memory)

- Kun aktiv ved lav (0)  $\overline{CS}$  vha. NAND-gate

$$\overline{CS} = \overline{A_{12} \cdot A_{13} \cdot A_{14} \cdot A_{15}}$$

Chip er aktiv, når alle A12-A15 er høje (1).

- RAM er i toppen af hukommelsen.

FFFF	RAM	<b>1111</b> 1111 1111 1111
F000		<b>1111</b> 0000 0000 0000
FFFF		1110 1111 1111 1111
0100	ROM	0000 0001 0000 0000
00FF		<b>0000 0000</b> 1111 1111
0000		<b>0000 0000</b> 0000 0000

# Hukommelsesinterface: 8-bit

## Memory map

ROM (read-only memory)

- Kun aktiv ved lav (0)  $\overline{CS}$  vha. OR-gate

$$\overline{CS} = A_8 + A_9 + \dots + A_{14} + A_{15}$$

Chip er aktiv, når alle A8-A15 er lave (0).

- ROM er i bunden af hukommelsen.

FFFF	RAM	<b>1111</b> 1111 1111 1111
F000		<b>1111</b> 0000 0000 0000
FFFF		1110 1111 1111 1111
0100	ROM	0000 0001 0000 0000
00FF		<b>0000 0000</b> 1111 1111
0000		<b>0000 0000</b> 0000 0000

# Hukommelsesinterface: 8-bit

## Memory map

ROM (read-only memory)

- Kun aktiv ved lav (0)  $\overline{CS}$  vha. OR-gate

$$\overline{CS} = A_8 + A_9 + \dots + A_{14} + A_{15}$$

Chip er aktiv, når alle A8-A15 er lave (0).

- ROM er i bunden af hukommelsen.

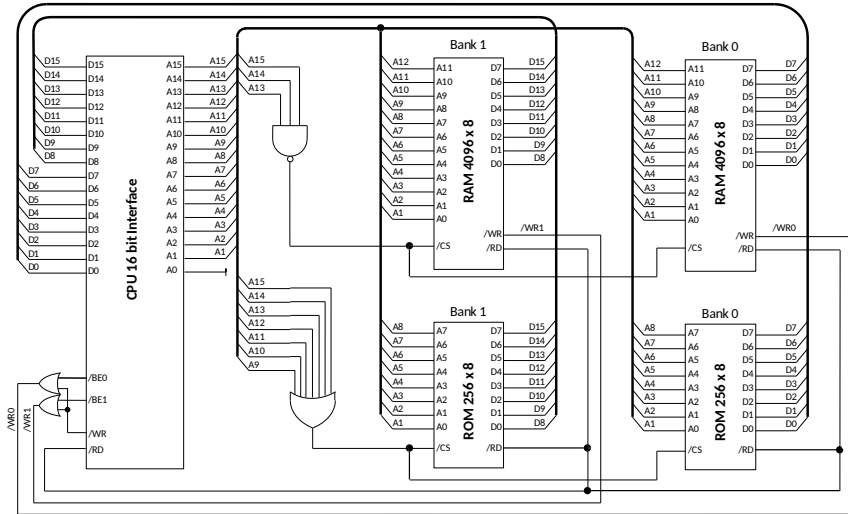
$\overline{WR}$  og  $\overline{RD}$  bestemmer om hukommelsen bliver hhv. læst eller skrevet.

$\overline{WR}$  går ind i RAM og  $\overline{RD}$  går i både RAM og ROM.

FFFF	RAM	1111 1111 1111 1111
F000		1111 0000 0000 0000
FFFF		1110 1111 1111 1111
0100	ROM	0000 0001 0000 0000
00FF		0000 0000 1111 1111
0000		0000 0000 0000 0000



# Hukommelsesinterface: 16-bit



# Hukommelsesinterface: 16-bit

## Memory map

A0 er ikke i brug!

$\overline{BE0}$  og  $\overline{BE1}$  er *bank-select* og bruges sammen med  $\overline{WR}$ .

Kan bruges til at skrive til den ene eller den anden bank.

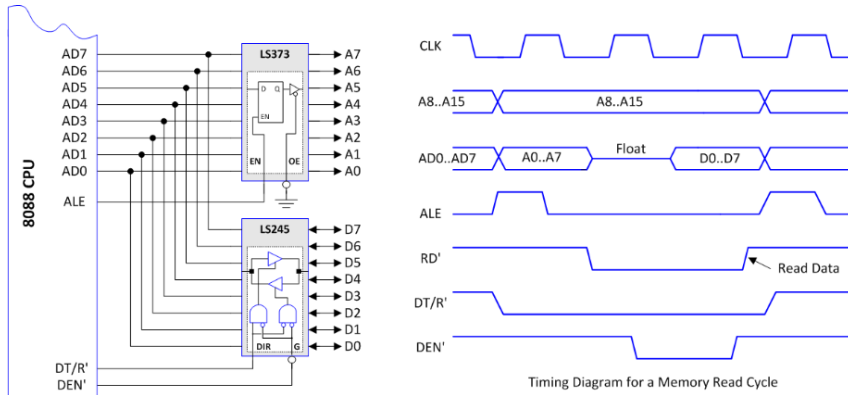
Bank 1			Bank 0		
FFFF	RAM	1111 1111 1111 1111	FFFE	RAM	1111 1111 1111 1110
E001		1110 0000 0000 0001	E000		1110 0000 0000 0000
CFFF		1101 1111 1111 1111	CFFE		1101 1111 1111 1110
0201	ROM	0000 0010 0000 0001	0200	ROM	0000 0010 0000 0000
01FF		0000 0000 1111 1111	01FE		0000 0000 1111 1110
0001		0000 0000 0000 0001	0000		0000 0000 0000 0000

# Hukommelsesinterface

Mange CPUer har *ikke* separate adresse-/data-ben

I den tidlige udvikling af CPUen har man haft et begrænset antal ben til rådighed.

Adresse bliver latched først, og så kan man skrive/læse data.



Hukommelsesinterface

Wait State

Intel 8088

ATmega128L

Adresse-demultiplexer

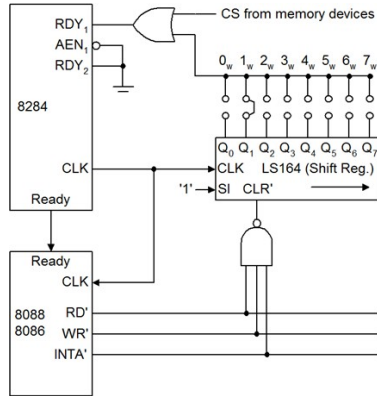
ATmega128L

Opsummering

Referencer

# Wait State – Intel 8088

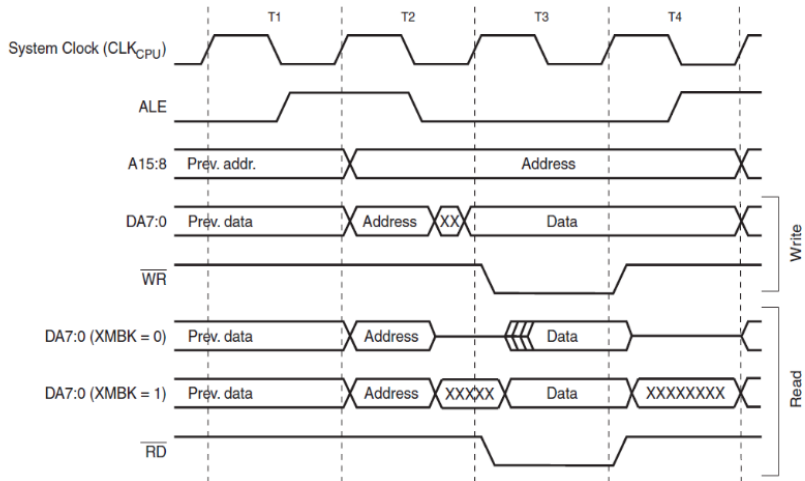
Nogle chipsets (hukommelse) kan have brug for ekstra tid til at finde dataen. Derfor kan der være brug for at **vente**, f.eks. vha. et wait-state generator-kredsløb.



# ATmega128L – External Memory

No wait state

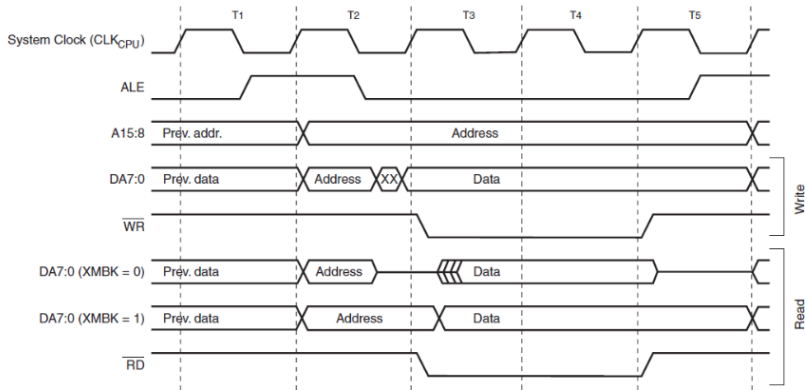
**Figure 13.** External Data Memory Cycles without Wait-state (SRWn1=0 and SRWn0=0)



# ATmega128L – External Memory

One wait state

**Figure 14.** External Data Memory Cycles with  $SRWn1 = 0$  and  $SRWn0 = 1$ <sup>(1)</sup>



Hukommelsesinterface

Wait State

Adresse-demultiplexer

ATmega128L

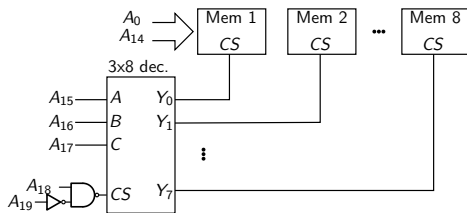
Opsummering

Referencer



# Adresse-demultiplexer

74xx137 3 to 8 decoder



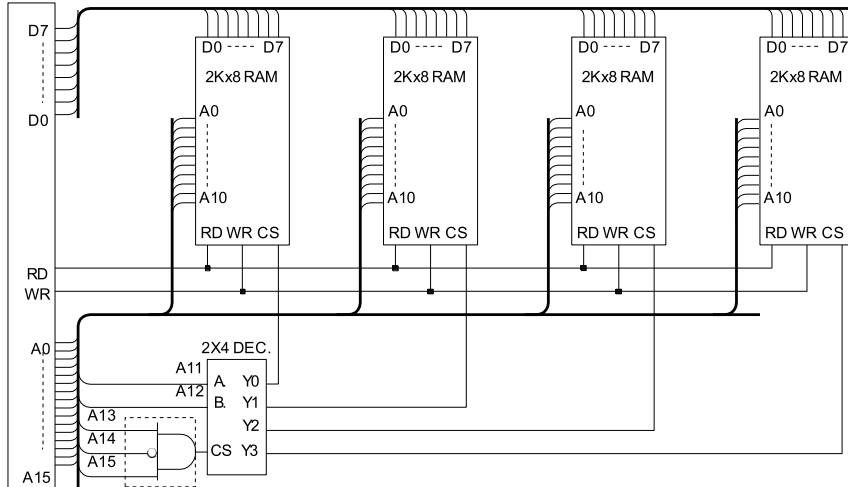
A <sub>19</sub>	A <sub>18</sub>	A <sub>17</sub>	A <sub>16</sub>	A <sub>15</sub>	A <sub>14</sub>	...	A <sub>0</sub>	Memory space	
0	1	0	0	0	0	...	0	0x40000	MEM1
0	1	0	0	0	1	...	1	0x47FFF	
0	1	0	0	1	0	...	0	0x48000	MEM2
0	1	0	0	1	1	...	1	0x4FFFF	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	1	1	1	1	0	...	0	0x78000	MEM8
0	1	1	1	1	1	...	1	0x7FFFF	

Bruges, når forskellige hukommelses-chipsets bruges til samme adresse-blok.

- Y-outputs bliver forbundet med CS til hukommelsen.
- A<sub>0</sub>–A<sub>14</sub> bruges til at adressere hukommelsen.
- A<sub>15</sub>–A<sub>16</sub> bruges til at adressere chippen. Understøtter  $2^3 = 8$  chips.
- A<sub>18</sub>–A<sub>19</sub> bruges til at vælge 74xx138 med CS.

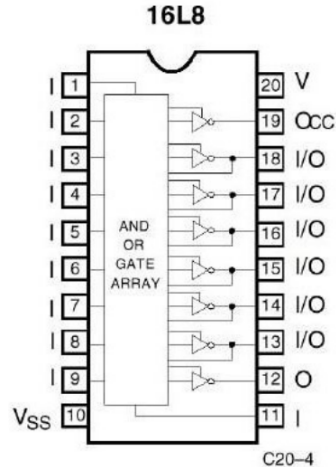
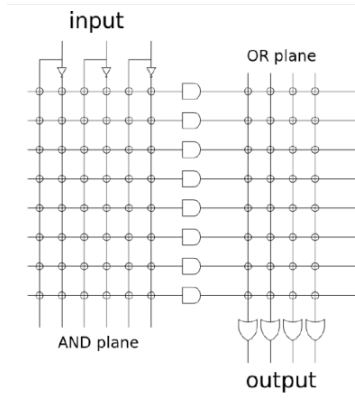
# Adresse-demultiplexer

74xx139 2 to 4 decoder



# Programmerbar Array-Logik

Adresse-demultiplexer kan programmeres vha. Boolsk algebra.



Hukommelsesinterface

Wait State

Adresse-demultiplexer

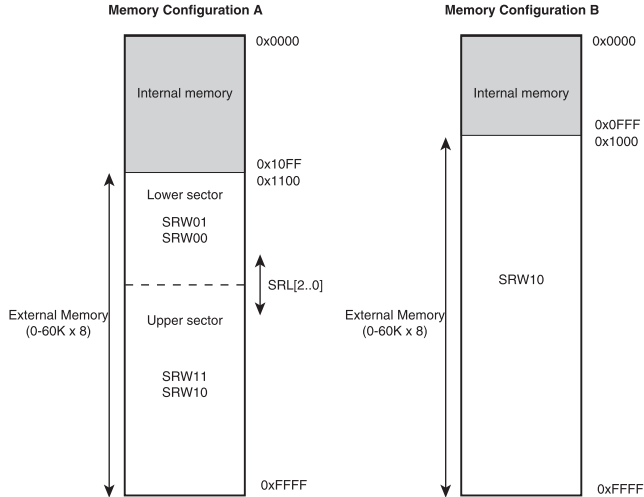
ATmega128L

Opsummering

Referencer

# ATmega128L – 8-bit microcontroller

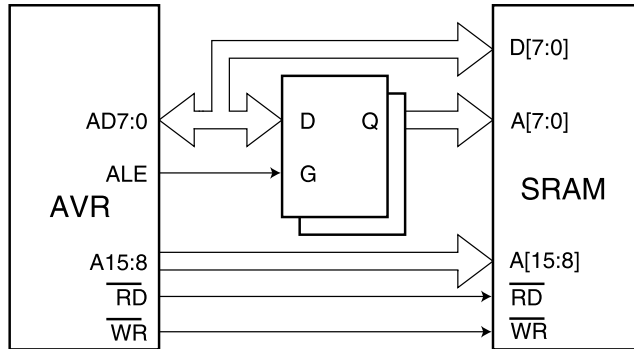
## Ekstern Hukommelse



# ATmega128L – Ekstern Hukommelse

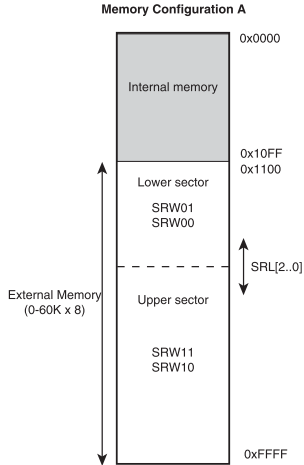
Multiplekset adresse-/data-bus

Forbundet vha. en latch, f.eks. en octal latch 74x573.



# ATmega128L – Ekstern Hukommelse

## Sektor Split



**Table 3.** Sector limits with different settings of SRL2..0

SRL2	SRL1	SRL0	Sector Limits
0	0	0	Lower sector = N/A Upper sector = 0x1100 - 0xFFFF
0	0	1	Lower sector = 0x1100 - 0x1FFF Upper sector = 0x2000 - 0xFFFF
0	1	0	Lower sector = 0x1100 - 0x3FFF Upper sector = 0x4000 - 0xFFFF
0	1	1	Lower sector = 0x1100 - 0x5FFF Upper sector = 0x6000 - 0xFFFF
1	0	0	Lower sector = 0x1100 - 0x7FFF Upper sector = 0x8000 - 0xFFFF
1	0	1	Lower sector = 0x1100 - 0x9FFF Upper sector = 0xA000 - 0xFFFF
1	1	0	Lower sector = 0x1100 - 0xBFFF Upper sector = 0xC000 - 0xFFFF
1	1	1	Lower sector = 0x1100 - 0xDFFF Upper sector = 0xE000 - 0xFFFF

Konfigurerbar sektor-split med forskellige wait-state instillinger.

SRW11 og SRW10 i MCU Control Register (MCUCR) styrer antallet af wait-states for den øvre del af hukommelsen.

SRW01 og SRW00 i External Memory Control Register A (XMCRA) styrer antallet af wait-states for den nederste del af hukommelsen.

**Table 4.** Wait States<sup>(1)</sup>

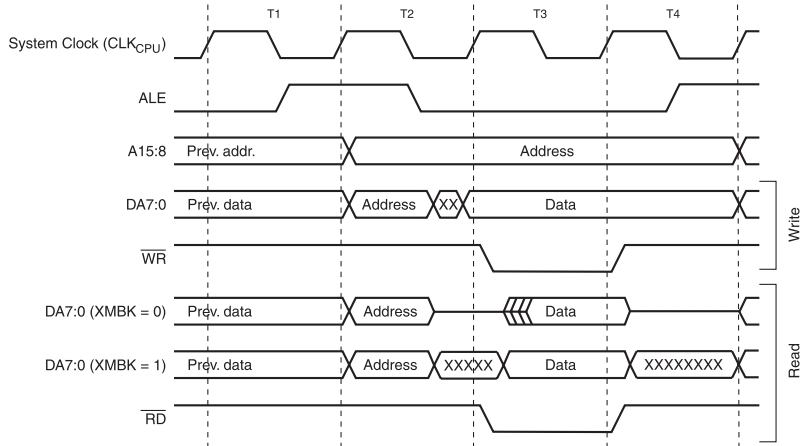
SRWn1	SRWn0	Wait States
0	0	No wait-states
0	1	Wait one cycle during read/write strobe
1	0	Wait two cycles during read/write strobe
1	1	Wait two cycles during read/write and wait one cycle before driving out new address



# ATmega128L – Ekstern Hukommelse

Bus-cyklus: Intet wait-state

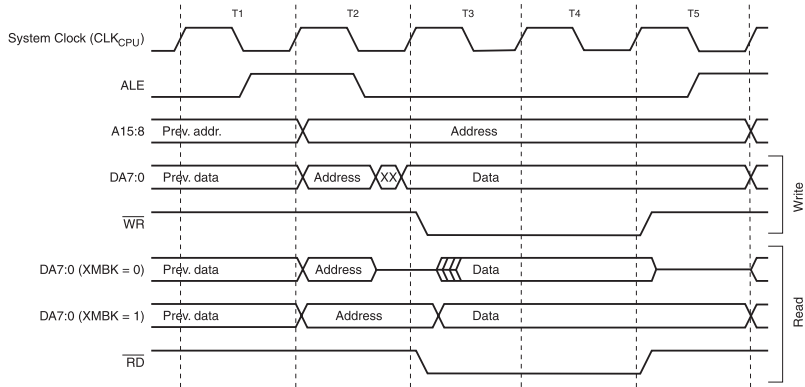
**Figure 13.** External Data Memory Cycles without Wait-state ( $SRWn1=0$  and  $SRWn0=0$ )



# ATmega128L – Ekstern Hukommelse

Bus-cyklus: Et wait-state

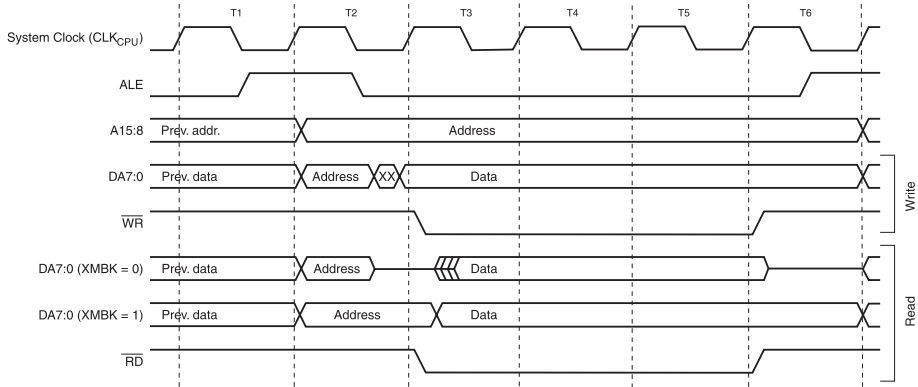
**Figure 14.** External Data Memory Cycles with  $SRWn1 = 0$  and  $SRWn0 = 1$ <sup>(1)</sup>



# ATmega128L – Ekstern Hukommelse

Bus-cyklus: To wait-state

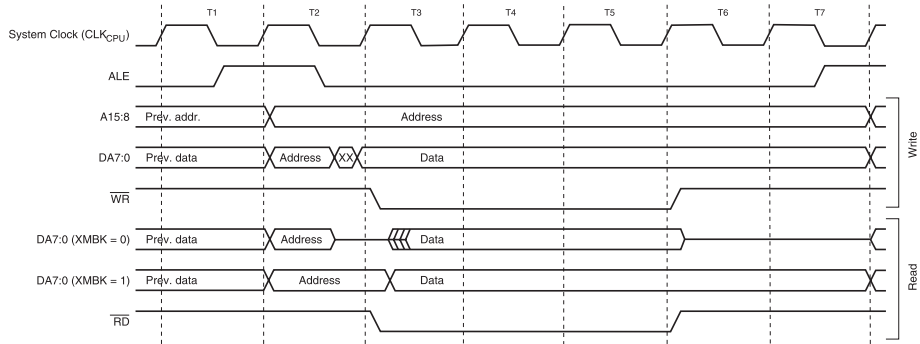
**Figure 15.** External Data Memory Cycles with  $SRWn1 = 1$  and  $SRWn0 = 0^{(1)}$



# ATmega128L – Ekstern Hukommelse

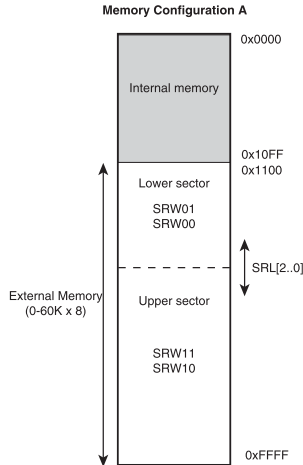
Bus-cyklus: To wait state + wait state før næste cyklus

**Figure 16.** External Data Memory Cycles with  $SRWn1 = 1$  and  $SRWn0 = 1$ <sup>(1)</sup>



# ATmega128L – Ekstern Hukommelse

## Adresse-setup



Hvis ikke al hukommelsen skal bruges, kan nogle af Port C-benene frigøres og bruges til andre formål.

**Table 5.** Port C Pins Released as Normal Port Pins when the External Memory is Enabled

XMM2	XMM1	XMM0	# Bits for External Memory Address	Released Port Pins
0	0	0	8 (Full 60 KB space)	None
0	0	1	7	PC7
0	1	0	6	PC7 - PC6
0	1	1	5	PC7 - PC5
1	0	0	4	PC7 - PC4
1	0	1	3	PC7 - PC3
1	1	0	2	PC7 - PC2
1	1	1	No Address high bits	Full Port C

Hukommelsesinterface

Wait State

Adresse-demultiplexer

ATmega128L

Opsummering

Referencer

- Hukommelsesinterface, herunder 8- og 16-bit.
- Nogle chips kan være langsomme til at finde data, så derfor kan et **wait state**-kredsløb være nødvendigt.
- Adressering af hukommelse vha. demultiplexer.
- Bus-cyklus.
- Forbindelse af ekstern hukommelse til ATmega128L.

- [1] Intel, **8088, 8-bit HMOS Microprocessor, 8088/8088-2**, 1979.
- [2] B. Brey, „The Intel Microprocessors, 8086/8088, 80186, 80286, 80386, and 80486, Architecture, Programming, and Interfacing, New York, NY: Merrill, 1994,“, 2009.
- [3] Atmel, **ATmega128L Datasheet**, Rev. 2467XS-AVR-06/11, 2011.
- [4] A. S. Tanenbaum, T. Austin og B. Chandavarkar, **Structured computer organization**, eng, 6. edition. International edition. Boston, Mass: Pearson, 2013.



- Lav en adresse-demultiplexer som beskrevet på slide 23.  
Benyt basis kredse frem for dekoder-blokken.
- Implementer din egen 4x3 hukommelseskreds vha D-flip-flop.  
Se figur 3-28 på side 176 i [4].