

# Maskinorienterad Programmering 2011/2012

MC68HC12,  
Arbetsbok för MC12  
CPU12 Reference Guide

Ur innehållet:  
Programmerarens bild  
Översikt, "single-chip-computer" DG256  
(Exempel)

## Instruktionsuppsättning

"ISA" – Instruction Set Architecture

- ⊕ Vilka operationer kan utföras ?
  - Instruktionsgrupper
- ⊕ Hur lagras operanderna förutom i minnet ?
  - Korttidslagring
- ⊕ Hur nås operander i minnet?
  - Adresseringssätt
- ⊕ Vilka typer/storlekar av operander kan hanteras ?
  - Generella/speciella register, registerstorlek

## Programmerarens bild – datatyper/storlek

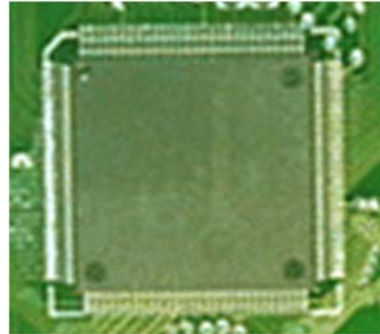
|              | char<br>(8) | short<br>int<br>(16) | long<br>int<br>(32) | long<br>int<br>(64) | floating<br>point<br>(IEEE) | pointers  |
|--------------|-------------|----------------------|---------------------|---------------------|-----------------------------|-----------|
| 68HCS12      | X           | X                    |                     |                     |                             | 16/20 bit |
| Coldfire V1  | X           | X                    | X                   |                     |                             | 32 bit    |
| Coldfire V4  | X           | X                    | X                   |                     | X                           | 32 bit    |
| PowerPC      | X           | X                    | X                   |                     | X                           | 32 bit    |
| PowerPC (64) | X           | X                    |                     | X                   | X                           | 64 bit    |
| 8086         | X           | X                    |                     |                     |                             | 16/20 bit |
| 80386        | X           | X                    | X                   |                     |                             | 32 bit    |
| 80486        | X           | X                    | X                   |                     | X                           | 32 bit    |
| X86-32       | X           | X                    | X                   |                     | X                           | 32 bit    |
| X86-64       | X           | X                    |                     | X                   | X                           | 64 bit    |

## Programmerarens bild – adresserbart minne

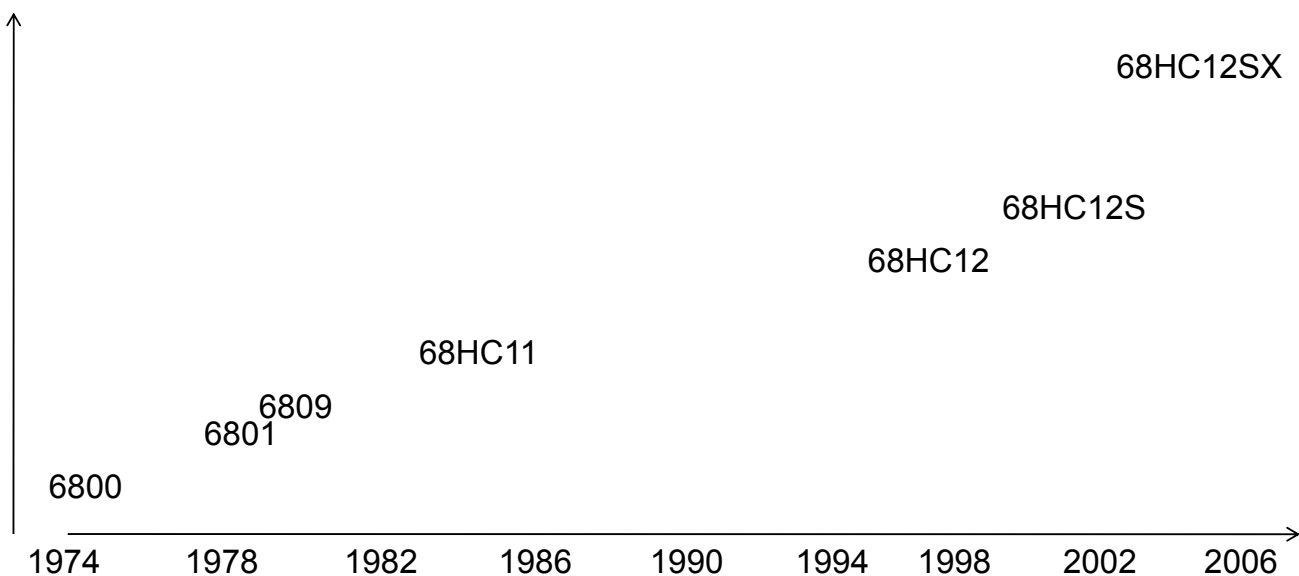
| ADRESSBUSS | RANDOM ACCESS                                                                                                 |
|------------|---------------------------------------------------------------------------------------------------------------|
| 16 bitar   | $2^{16} = 65\,536 \text{ byte} = 64 \text{ kbyte}$                                                            |
| 20 bitar   | $2^{20} = 1\,048\,576 \text{ byte} = 1\,024 \text{ kbyte} = 1 \text{ Mbyte}$                                  |
| 24 bitar   | $2^{24} = 16\,777\,216 \text{ byte} = 16\,384 \text{ kbyte} = 16 \text{ MByte}$                               |
| 32 bitar   | $2^{32} = 4\,294\,967\,296 \text{ byte} = 4\,194\,304 \text{ kbyte} = 4\,096 \text{ Mbyte} = 4 \text{ Gbyte}$ |
| 64 bitar   | $2^{64} = 1,844674407 \cdot 10^{19} \text{ byte} = 16 \text{ Ebyte}$                                          |

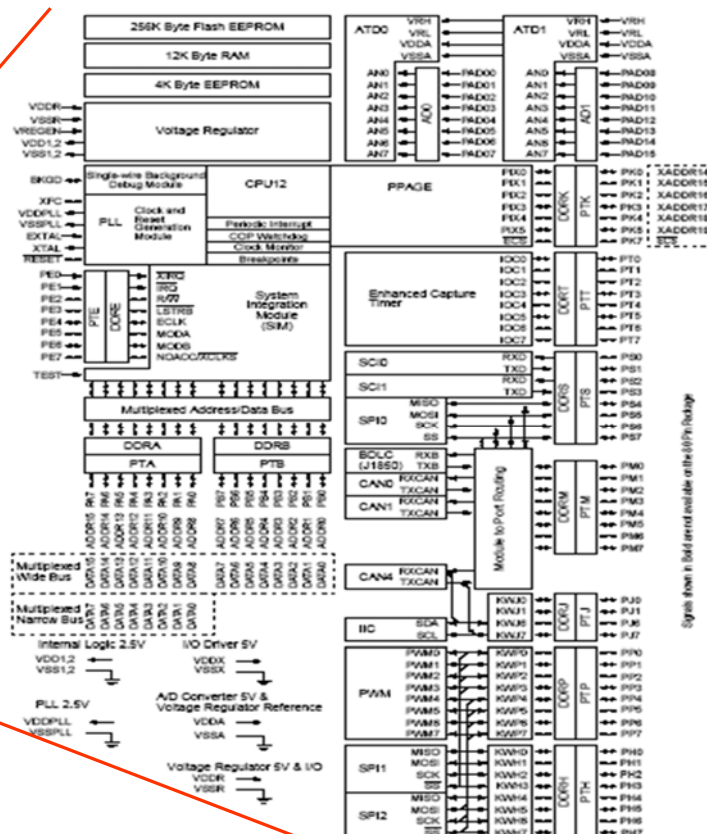
# Freescal 68HCS12

- HCS12 adressrum, IO och minne
- CPU12, klockor och räknare
- "Random Access"- Minne
  - RWM, FLASH, EEPROM
- Periferienheter
  - Parallell Input/Output:
  - Seriell kommunikation
  - AD
  - PWM



# Historik

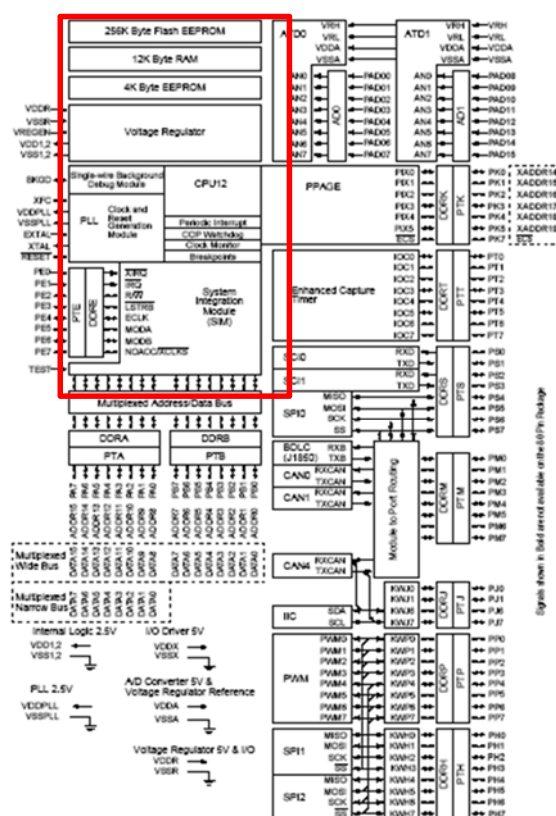




**Keywords:** *Personality*, *Self-esteem*, *Depression*

7

The block diagram illustrates the internal architecture of the AT91SAM7S64 microcontroller. At the top, three memory blocks are shown: 256K Byte Flash EEPROM, 12K Byte RAM, and 4K Byte EEPROM. Below these is the Voltage Regulator, which receives power inputs VDDR, VSSR, VREGEN, VDD1,2, and VSS1,2. The Single-wire Background Debug Module (BKGD) is connected to the CPU12 core. The PLL (Clock and Reset Generation Module) is connected to the CPU12 core and provides clock signals to the PTE and DDRE registers. The CPU12 core includes a Periodic Interrupt, COP Watchdog, Clock Monitor, and Breakpoints. The PTE and DDRE registers are connected to the PE0-PE7 pins. The TEST pin is connected to the TEST input. The System Integration Module (SIM) is connected to the CPU12 core and provides signals for XIRQ, IRQ, R/W, LSTRB, ECLK, MODA, MODB, and NOACC/XCLKS.



Signs of an In-Bed are not available on the 6.0 Pin Package

8

## HCS12DG256, "core"

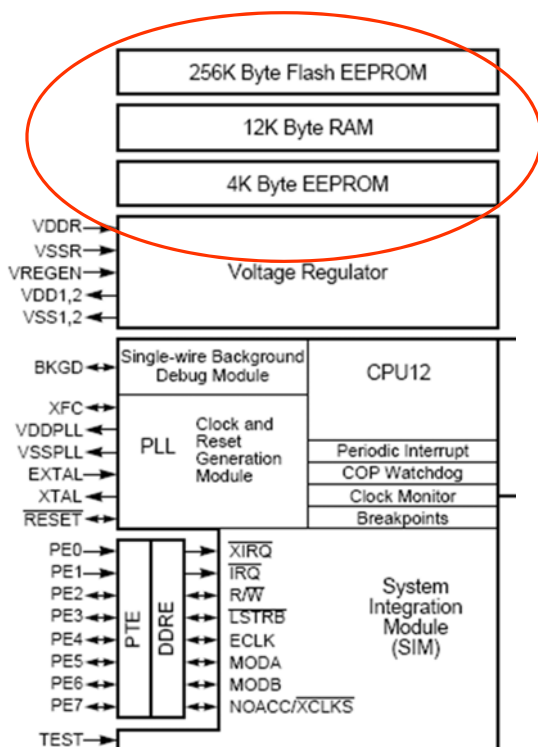
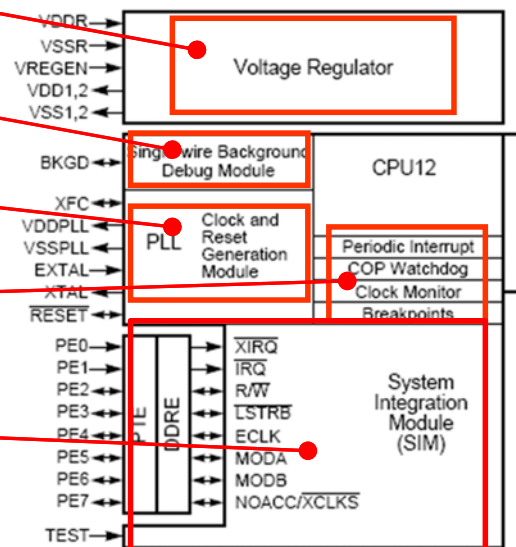
Spänningsregulatorer (flera olika spänningar används internt)

"Background Debug Mode" för test/avlusning

En kristall utgör bas för alla klockfrekvenser i systemet

Realtidsklocka och andra klockfunktioner

Programmerbara funktioner



## Primärminne

Icke flyktigt minne

Upp till 256 Kbyte i "minnesbankar"

48 kB utan användning av "bankar"

4 kB EEPROM

Flyktigt minne

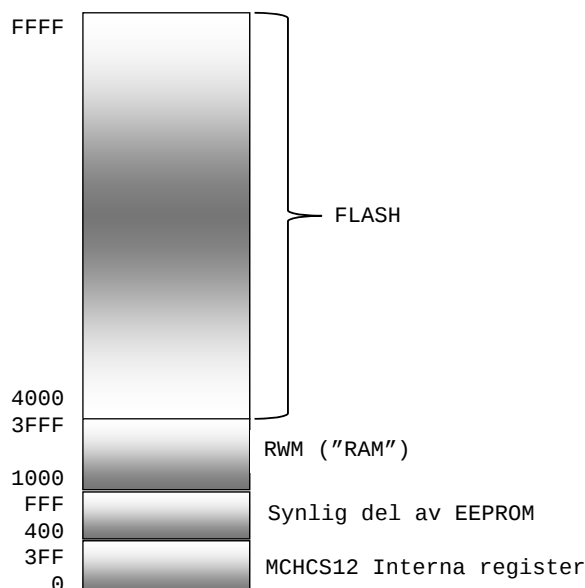
12 kB RAM (=RWM)

## EXEMPEL, linjärt adressrum

256K Byte Flash EEPROM

12K Byte RAM

4K Byte EEPROM

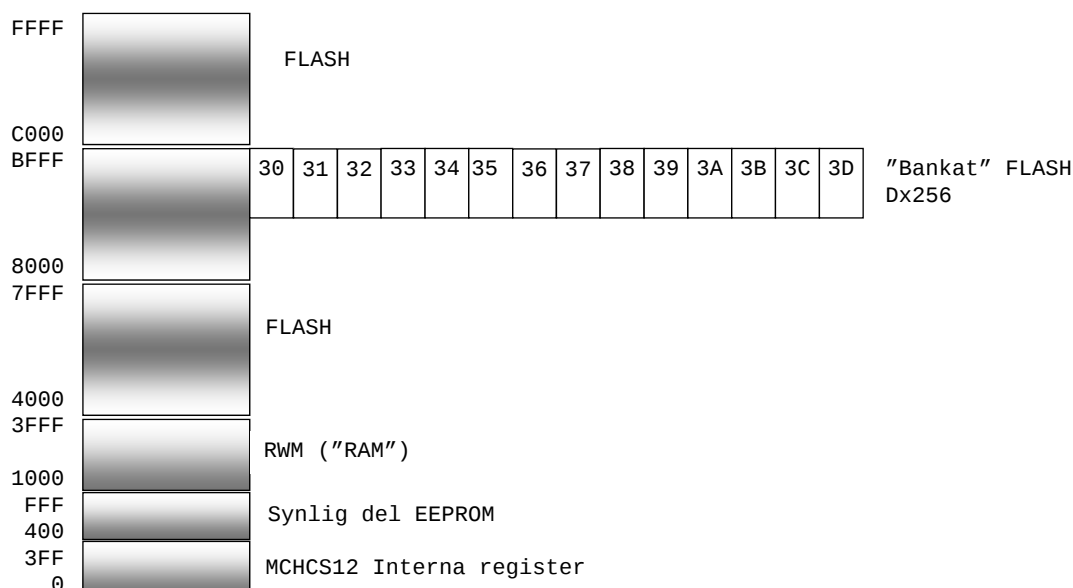


## EXEMPEL, "bankat" adressrum

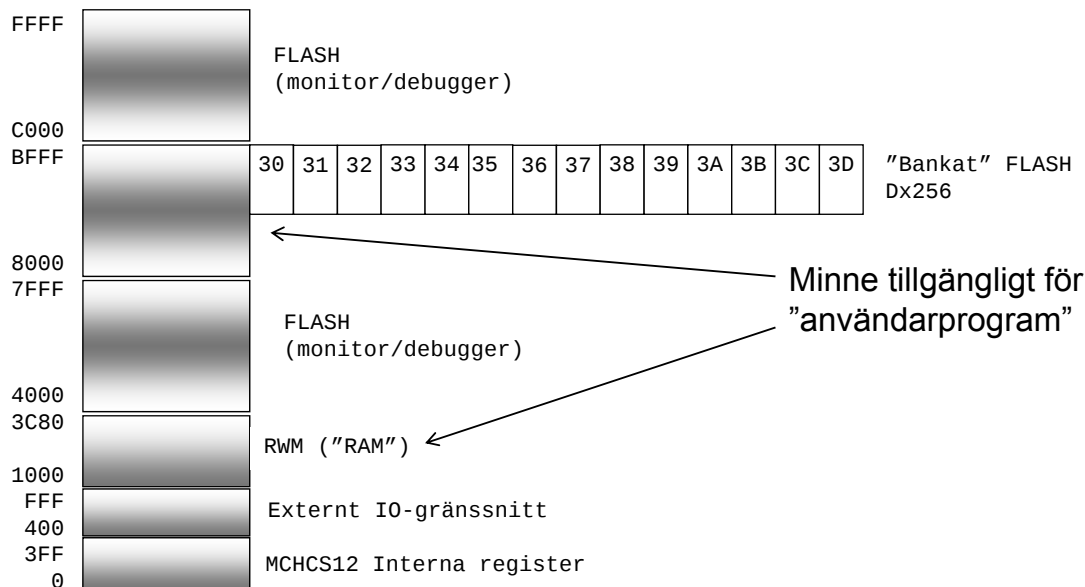
256K Byte Flash EEPROM

12K Byte RAM

4K Byte EEPROM



# EXEMPEL, i laborationsdator MC12



|                        |
|------------------------|
| 256K Byte Flash EEPROM |
| 12K Byte RAM           |
| 4K Byte EEPROM         |

## Periferikretsar i HCS12DG256

AD – Analog till Digital omvandling

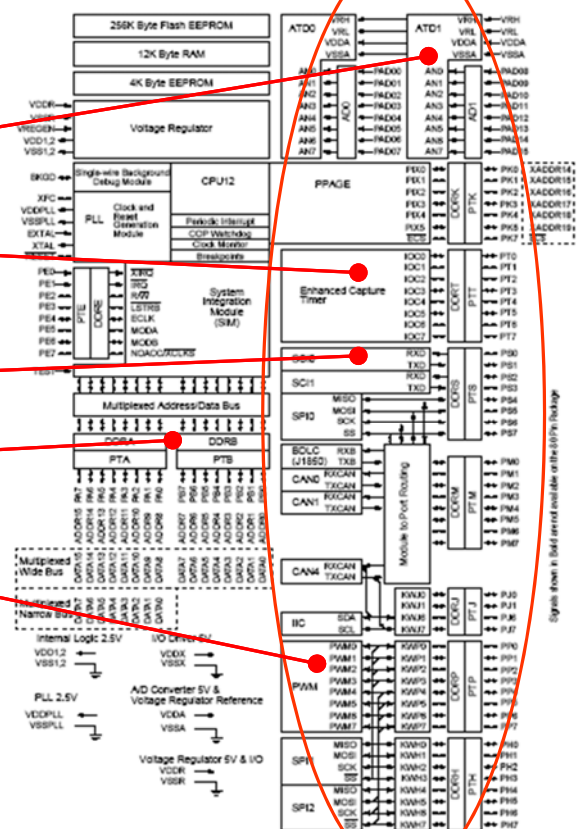
ECT- Räknarkretsar för noggrann tidmätning

SCI – Asynkron seriekommunikation

Parallell In-Utmatning

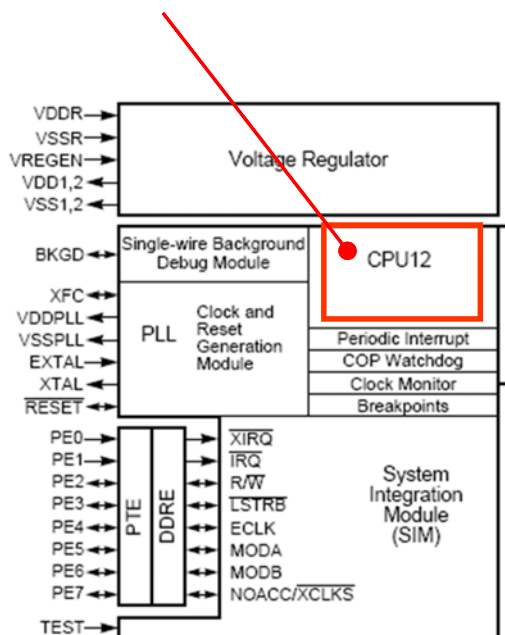
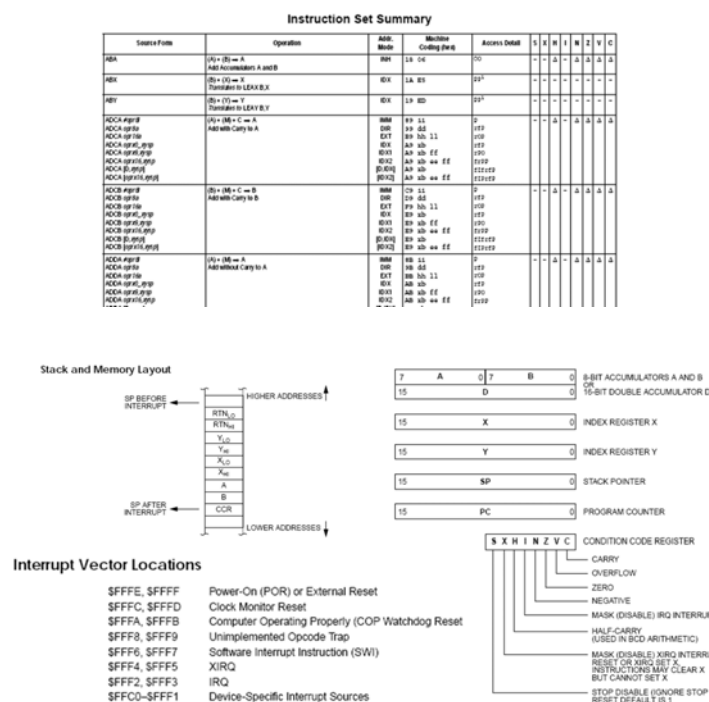
PWM – Pulsbreddsmodulering

Etc...



## HCS12DG256, "core"

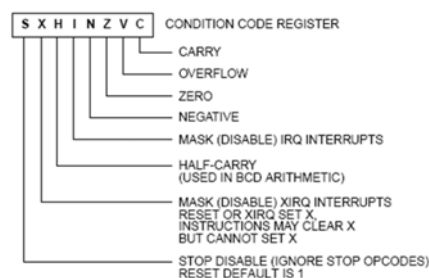
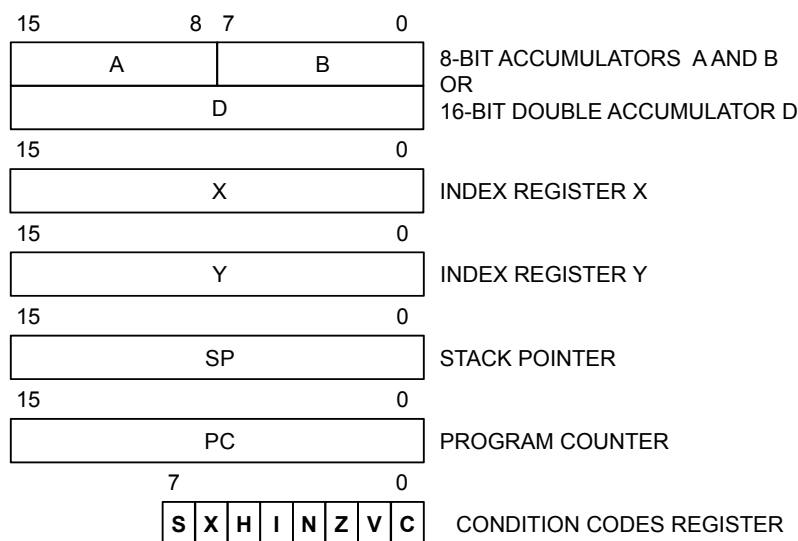
Centralenhet CPU12



## Introduktion till Freescale MC68HCS12

15

## Registeruppsättning CPU12



## Introduktion till Freescale MC68HCS12

16

## Adresseringssätt

Vi känner igen de flesta adresseringssätten i från FLEX.

Indexerade adresseringssätt kan även användas med register X, Y och SP ibland också med PC (PC-relativt)

Nytt här är också "Indirekt adressering"

| Addressing Mode                         | Source Format                          | Abbreviation | Description                                                                                                          |
|-----------------------------------------|----------------------------------------|--------------|----------------------------------------------------------------------------------------------------------------------|
| Inherent                                | INST (no externally supplied operands) | INH          | Operands (if any) are in CPU registers                                                                               |
| Immediate                               | INST #opr8i<br>or<br>INST #opr16i      | IMM          | Operand is included in instruction stream 8- or 16-bit size implied by context                                       |
| Direct                                  | INST opr8a                             | DIR          | Operand is the lower 8-bits of an address in the range \$0000 – \$00FF                                               |
| Extended                                | INST opr16a                            | EXT          | Operand is a 16-bit address                                                                                          |
| Relative                                | INST rel8<br>or<br>INST rel16          | REL          | An 8-bit or 16-bit relative offset from the current pc is supplied in the instruction                                |
| Indexed (5-bit offset)                  | INST oprx5,xysp                        | IDX          | 5-bit signed constant offset from x, y, sp, or pc                                                                    |
| Indexed (pre-decrement)                 | INST oprx3,-xys                        | IDX          | Auto pre-decrement x, y, or sp by 1 ~ 8                                                                              |
| Indexed (pre-increment)                 | INST oprx3,+xys                        | IDX          | Auto pre-increment x, y, or sp by 1 ~ 8                                                                              |
| Indexed (post-decrement)                | INST oprx3,xys-                        | IDX          | Auto post-decrement x, y, or sp by 1 ~ 8                                                                             |
| Indexed (post-increment)                | INST oprx3,xys+                        | IDX          | Auto post-increment x, y, or sp by 1 ~ 8                                                                             |
| Indexed (accumulator offset)            | INST abd,xysp                          | IDX          | Indexed with 8-bit (A or B) or 16-bit (D) accumulator offset from x, y, sp, or pc                                    |
| Indexed (9-bit offset)                  | INST oprx9,xysp                        | IDX1         | 9-bit signed constant offset from x, y, sp, or pc (lower 8-bits of offset in one extension byte)                     |
| Indexed (16-bit offset)                 | INST oprx16,xysp                       | IDX2         | 16-bit constant offset from x, y, sp, or pc (16-bit offset in two extension bytes)                                   |
| Indexed-Indirect (16-bit offset)        | INST [oprx16,xysp]                     | [IDX2]       | Pointer to operand is found at... 16-bit constant offset from x, y, sp, or pc (16-bit offset in two extension bytes) |
| Indexed-Indirect (D accumulator offset) | INST [D,xysp]                          | [D,IDX]      | Pointer to operand is found at... x, y, sp, or pc plus the value in D                                                |

## Inherent

| Source Form | Operation                               | Addr. Mode | Machine Coding (hex) | Access Detail | S | X | H | I | N | Z | V | C |
|-------------|-----------------------------------------|------------|----------------------|---------------|---|---|---|---|---|---|---|---|
| CBA         | (A) - (B)<br>Compare 8-Bit Accumulators | INH        | 18 17                | ∞             | - | - | - | - | Δ | Δ | Δ | Δ |

Maskinkod för instruktionen

Cykel för cykel beskrivning

Flaggpåverkan

## Omedelbar (Immediate) 8-bit/16-bit

|                    |                                 |         |             |         |   |   |   |   |   |   |   |   |
|--------------------|---------------------------------|---------|-------------|---------|---|---|---|---|---|---|---|---|
| <b>LDA #opr8i</b>  | (M) ⇒ A                         | IMM     | 86 11       | P       | - | - | - | - | Δ | Δ | 0 | - |
| LDA opr8a          | Load Accumulator A              | DIR     | 96 dd       | rFP     |   |   |   |   |   |   |   |   |
| LDA opr16a         |                                 | EXT     | B6 hh 11    | rOP     |   |   |   |   |   |   |   |   |
| LDA oprx0_xysp     |                                 | IDX     | A6 xb       | rFP     |   |   |   |   |   |   |   |   |
| LDA oprx9_xysp     |                                 | IDX1    | A6 xb ff    | rPO     |   |   |   |   |   |   |   |   |
| LDA oprx16_xysp    |                                 | IDX2    | A6 xb ee ff | frPP    |   |   |   |   |   |   |   |   |
| LDA [D, xysp]      |                                 | [D,IDX] | A6 xb       | fIfrrFP |   |   |   |   |   |   |   |   |
| LDA [opr16, xysp]  |                                 | [IDX2]  | A6 xb ee ff | fIDrFP  |   |   |   |   |   |   |   |   |
| <b>LDD #opr16i</b> | (M:M+1) ⇒ A:B                   | IMM     | CC jj kk    | OP      | - | - | - | - | Δ | Δ | 0 | - |
| LDD opr8a          | Load Double Accumulator D (A:B) | DIR     | DC dd       | rFP     |   |   |   |   |   |   |   |   |
| LDD opr16a         |                                 | EXT     | FC hh 11    | rOP     |   |   |   |   |   |   |   |   |
| LDD oprx0_xysp     |                                 | IDX     | EC xb       | rFP     |   |   |   |   |   |   |   |   |
| LDD oprx9_xysp     |                                 | IDX1    | EC xb ff    | rPO     |   |   |   |   |   |   |   |   |
| LDD oprx16_xysp    |                                 | IDX2    | EC xb ee ff | frPP    |   |   |   |   |   |   |   |   |
| LDD [D, xysp]      |                                 | [D,IDX] | EC xb       | fIfrrFP |   |   |   |   |   |   |   |   |
| LDD [opr16, xysp]  |                                 | [IDX2]  | EC xb ee ff | fIDrFP  |   |   |   |   |   |   |   |   |

*opr8i*, 8-bitars konstant om 8-bitars register

*Opr16i*, 16-bitars konstant om 16-bitars register

## Direkt (Direct Page) Absolut (Extended)

|                    |                                 |         |             |         |   |   |   |   |   |   |   |   |
|--------------------|---------------------------------|---------|-------------|---------|---|---|---|---|---|---|---|---|
| <b>LDA #opr8i</b>  | (M) ⇒ A                         | IMM     | 86 11       | P       | - | - | - | - | Δ | Δ | 0 | - |
| <b>LDA opr8a</b>   | Load Accumulator A              | DIR     | 96 dd       | rFP     |   |   |   |   |   |   |   |   |
| <b>LDA opr16a</b>  |                                 | EXT     | B6 hh 11    | rOP     |   |   |   |   |   |   |   |   |
| LDA oprx0_xysp     |                                 | IDX     | A6 xb       | rFP     |   |   |   |   |   |   |   |   |
| LDA oprx9_xysp     |                                 | IDX1    | A6 xb ff    | rPO     |   |   |   |   |   |   |   |   |
| LDA oprx16_xysp    |                                 | IDX2    | A6 xb ee ff | frPP    |   |   |   |   |   |   |   |   |
| LDA [D, xysp]      |                                 | [D,IDX] | A6 xb       | fIfrrFP |   |   |   |   |   |   |   |   |
| LDA [opr16, xysp]  |                                 | [IDX2]  | A6 xb ee ff | fIDrFP  |   |   |   |   |   |   |   |   |
| <b>LDD #opr16i</b> | (M:M+1) ⇒ A:B                   | IMM     | CC jj kk    | OP      | - | - | - | - | Δ | Δ | 0 | - |
| <b>LDD opr8a</b>   | Load Double Accumulator D (A:B) | DIR     | DC dd       | rFP     |   |   |   |   |   |   |   |   |
| <b>LDD opr16a</b>  |                                 | EXT     | FC hh 11    | rOP     |   |   |   |   |   |   |   |   |
| LDD oprx0_xysp     |                                 | IDX     | EC xb       | rFP     |   |   |   |   |   |   |   |   |
| LDD oprx9_xysp     |                                 | IDX1    | EC xb ff    | rPO     |   |   |   |   |   |   |   |   |
| LDD oprx16_xysp    |                                 | IDX2    | EC xb ee ff | frPP    |   |   |   |   |   |   |   |   |
| LDD [D, xysp]      |                                 | [D,IDX] | EC xb       | fIfrrFP |   |   |   |   |   |   |   |   |
| LDD [opr16, xysp]  |                                 | [IDX2]  | EC xb ee ff | fIDrFP  |   |   |   |   |   |   |   |   |

*opr16a*, kan adressera hela adressintervallet 0000-FFFF

*opr8a*, kan enbart adressera intervallet 0000-00FF, anger minst signifikant byte av adressen

## PC-relativ ("BRANCH"-instruktioner)

- 8-bitars offset (-128..127)
- 9-bitars offset (-256..255)
- 16-bitars offset (-32768..32767)

|                          |                                                                                                                                                                                    |                |             |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-------------|
| BRA <i>rel8</i>          | Branch Always (if 1 = 1)                                                                                                                                                           | REL            | 20 rr       |
| IBEQ <i>abdxys, rel9</i> | (cntr) + 1 $\Rightarrow$ cntr<br>If (cntr) = 0, then Branch<br>else Continue to next instruction<br>Increment Counter and Branch if = 0<br>(cntr = A, B, D, X, Y, or SP)           | REL<br>(9-bit) | 04 1b rr    |
| IBNE <i>abdxys, rel9</i> | (cntr) + 1 $\Rightarrow$ cntr<br>if (cntr) not = 0, then Branch;<br>else Continue to next instruction<br>Increment Counter and Branch if $\neq$ 0<br>(cntr = A, B, D, X, Y, or SP) | REL<br>(9-bit) | 04 1b rr    |
| LBCC <i>rel16</i>        | Long Branch if Carry Clear (if C = 0)                                                                                                                                              | REL            | 18 24 qq rr |

## Introduktion till Freescale MC68HCS12

21

## Indexerade adresseringssätt:

- Register relativ, konstant offset

|                               |                     |
|-------------------------------|---------------------|
| LDAA # <i>opr8i</i>           | (M) $\Rightarrow$ A |
| LDAA <i>opr8a</i>             | Load Accumulator A  |
| LDAA <i>opr16a</i>            |                     |
| LDAA <i>opr0_xysp</i>         |                     |
| LDAA <i>opr9_xysp</i>         |                     |
| LDAA <i>opr16_xysp</i>        |                     |
| LDAA [D, <i>xysp</i> ]        |                     |
| LDAA [D, <i>opr16_xvsol</i> ] |                     |

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <i>opr0_xysp</i> | Indexed addressing postbyte code:                           |
| <i>opr3,-xys</i> | Predecrement X or Y or SP by 1...8                          |
| <i>opr3,+xys</i> | Preincrement X or Y or SP by 1...8                          |
| <i>opr3,xys-</i> | Postdecrement X or Y or SP by 1...8                         |
| <i>opr3,xys+</i> | Postincrement X or Y or SP by 1...8                         |
| <i>opr5_xysp</i> | 5-bit constant offset from X or Y or SP or PC               |
| <i>abd_xysp</i>  | Accumulator A or B or D offset from X or Y or SP or PC      |
| <i>opr3</i>      | Any positive integer 1...8 for pre/post increment/decrement |
| <i>opr5</i>      | Any value in the range -16...+15                            |
| <i>opr9</i>      | Any value in the range -256...+255                          |
| <i>opr16</i>     | Any value in the range -32,768...65,535                     |

Basregister kan vara något av: X,Y,SP,PC

EXEMPEL:

```
LDAA 5, X
STAA 20, Y
LDAA sym, PC
STA off, SP
...
```

Specialfall: n, PCR

LDAA sym, PCR

Antag PC pekar på nästa instruktion.

Operanden är här PC-sym, jfr offsetberäkning för "BRA"-instruktioner

Observera, ingen syntaktisk skillnad.  
Assemblator väljer effektivast kodning

Indexerade adresseringssätt:

- Auto pre- increment/decrement
- Auto post- increment/decrement

```
LDAA #opr8i
LDAA opr8a
LDAA opr16a
LDAA oprx0_xysp
LDAA oprx9_xysp
LDAA oprx16_xysp
LDAA [D,xysp]
LDAA [opr16.xvso]
```

(M) ⇒ A  
Load Accumulator A

*opr0\_xysp* — Indexed addressing postbyte code:

*opr3,-xys* Predecrement X or Y or SP by 1 . . . 8  
*opr3,+xys* Preincrement X or Y or SP by 1 . . . 8  
*opr3,xys-* Postdecrement X or Y or SP by 1 . . . 8  
*opr3,xys+* Postincrement X or Y or SP by 1 . . . 8

*opr5,xysp* 5-bit constant offset from X or Y or SP or PC

*abd,xysp* Accumulator A or B or D offset from X or Y or SP or PC

*opr3* — Any positive integer 1 . . . 8 for pre/post increment/decrement

*opr5* — Any value in the range -16 . . . +15

*opr9* — Any value in the range -256 . . . +255

*opr16* — Any value in the range -32,768 . . . 65,535

Basregister kan vara något av: X,Y,SP

EXEMPEL:

```
LDAA 1, -X
STAA 4, Y-
STAB 8, +SP
LDAB 7, SP+
...
```

Indexerade adresseringssätt:

- Register relativ, offset i ackumulator

```
LDAA #opr8i
LDAA opr8a
LDAA opr16a
LDAA oprx0_xysp
LDAA oprx9_xysp
LDAA oprx16_xysp
LDAA [D,xysp]
LDAA [opr16.xvso]
```

(M) ⇒ A  
Load Accumulator A

*opr0\_xysp* — Indexed addressing postbyte code:

*opr3,-xys* Predecrement X or Y or SP by 1 . . . 8  
*opr3,+xys* Preincrement X or Y or SP by 1 . . . 8  
*opr3,xys-* Postdecrement X or Y or SP by 1 . . . 8  
*opr3,xys+* Postincrement X or Y or SP by 1 . . . 8

*opr5,xysp* 5-bit constant offset from X or Y or SP or PC

*abd,xysp* Accumulator A or B or D offset from X or Y or SP or PC

*opr3* — Any positive integer 1 . . . 8 for pre/post increment/decrement

*opr5* — Any value in the range -16 . . . +15

*opr9* — Any value in the range -256 . . . +255

*opr16* — Any value in the range -32,768 . . . 65,535

Basregister kan vara något av: X,Y,SP,PC

EXEMPEL:

```
LDAA A, X
STAA B, Y
STAB D, SP
LDAB D, PC
...
```

## Indexerade adresseringssätt:

### Indirekt

|                   |                    |
|-------------------|--------------------|
| LDAA #opr8i       | (M) ⇒ A            |
| LDAA opr8a        | Load Accumulator A |
| LDAA opr16a       |                    |
| LDAA oprx0_xysp   |                    |
| LDAA oprx9_xysp   |                    |
| LDAA oprx16_xysp  |                    |
| LDAA [D,xysp]     |                    |
| LDAA [opr16,xvsp] |                    |

*opr0\_xysp* — Indexed addressing postbyte code:

- opr3,-xys* Predecrement X or Y or SP by 1...8
- opr3,+xys* Preincrement X or Y or SP by 1...8
- opr3,xys-* Postdecrement X or Y or SP by 1...8
- opr3,xys+* Postincrement X or Y or SP by 1...8
- opr5,xysp* 5-bit constant offset from X or Y or SP or PC
- abd,xysp* Accumulator A or B or D offset from X or Y or SP or PC

*opr3* — Any positive integer 1...8 for pre/post increment/decrement

*opr5* — Any value in the range -16...+15

*opr9* — Any value in the range -256...+255

*opr16* — Any value in the range -32,768...65,535

### EXEMPEL:

```
LDAA [D,X]
STAA [sym,PCR]
STAB [2,SP]
LDAB [D,Y]
...
```

## Instruktionsgrupper

### LOAD-instruktioner

| Mnemonic | Funktion                       | Operation                                              |
|----------|--------------------------------|--------------------------------------------------------|
| LDAA     | Load A                         | (M)→A                                                  |
| LDAB     | Load B                         | (M)→B                                                  |
| LDD      | Load D                         | (M:M+1) <sub>1</sub> →A:B                              |
| LDS      | Load SP                        | (M:M+1) <sub>1</sub> →SP <sub>H</sub> :SP <sub>L</sub> |
| LDX      | Load index register X          | (M:M+1) <sub>1</sub> →X <sub>H</sub> :X <sub>L</sub>   |
| LDY      | Load index register Y          | (M:M+1) <sub>1</sub> →Y <sub>H</sub> :Y <sub>L</sub>   |
| LEAS     | Load effective address into SP | Effective address→SP                                   |
| LEAX     | Load effective address into X  | Effective address→X                                    |
| LEAY     | Load effective address into Y  | Effective address→Y                                    |

### STORE-instruktioner

| Mnemonic | Funktion | Operation                               |
|----------|----------|-----------------------------------------|
| STAA     | Store A  | (A)→M                                   |
| STAB     | Store B  | (B)→M                                   |
| STD      | Store D  | (A)→M, (B)→M+1                          |
| STS      | Store SP | SP <sub>H</sub> :SP <sub>L</sub> →M:M+1 |
| STX      | Store X  | X <sub>H</sub> :X <sub>L</sub> →M:M+1   |
| STY      | Store Y  | Y <sub>H</sub> :Y <sub>L</sub> →M:M+1   |

### MOVE-instruktioner

| Mnemonic | Funktion            | Operation                                |
|----------|---------------------|------------------------------------------|
| MOVB     | Move byte (8 bitar) | (M <sub>1</sub> )→M <sub>2</sub>         |
| MOVW     | Move word (8 bitar) | (M:M+1) <sub>1</sub> →M:M+1 <sub>2</sub> |

### EXEMPEL: Kopiera byte

```
LDAB $3000
STAB $3001
eller
LDAA $3000
STAA $3001
eller
MOVB $3000,$3001
```

### EXEMPEL: Kopiera word

```
LDD $3000
STD $3001
eller
LDX $3000
STX $3001
eller
LDY $3000
STY $3001
eller
MOVW $3000,$3001
```

## Instruktioner för kopiering av registerinnehåll

| Mnemonic   | Funktion                                      | Operation                                                      |
|------------|-----------------------------------------------|----------------------------------------------------------------|
| TAB        | Transfer A to B<br>anm: Ekv. Med TFR A, B     | (A)→B                                                          |
| TAP        | Transfer A to CCR<br>anm: Ekv. Med TFR A, CCR | (A)→CCR                                                        |
| TBA        | Transfer B to A                               | (B)→A                                                          |
| <b>TFR</b> | <b>Transfer register to register</b>          | <b>(A,B,CCR,D,X,Y eller SP) →<br/>(A,B,CCR,D,X,Y eller SP)</b> |
| TPA        | Transfer CCR to A<br>anm: Ekv. Med TFR CCR, A | (CCR)→A                                                        |
| TSX        | Transfer SP to X<br>anm: Ekv. Med TFR SP, X   | (SP)→X                                                         |
| TSY        | Transfer SP to Y<br>anm: Ekv. Med TFR SP, Y   | (SP)→Y                                                         |
| TXS        | Transfer X to SP<br>anm: Ekv. Med TFR X, SP   | (X)→SP                                                         |
| TYS        | Transfer Y to SP<br>anm: Ekv. Med TFR Y, SP   | (Y)→SP                                                         |

← Använd denna

Övriga finns här av  
”kompatibilitetsskäl”

## Instruktioner för växling av registerinnehåll

| Mnemonic   | Funktion                                                  | Operation                                                      |
|------------|-----------------------------------------------------------|----------------------------------------------------------------|
| <b>EXG</b> | <b>Exchange register to register</b>                      | <b>(A,B,CCR,D,X,Y eller SP) ↔<br/>(A,B,CCR,D,X,Y eller SP)</b> |
| XGDY       | Exchange D with X<br>anm: Ekv. Med<br>EXG D, X - EXG X, D | (D) ↔ (X)                                                      |
| XGDY       | Exchange D with Y<br>anm: Ekv. Med<br>EXG D, Y - EXG Y, D | (D) ↔ (Y)                                                      |

← Använd denna

Övriga finns här av  
”kompatibilitetsskäl”

## Instruktion för teckenutvidgning

| Mnemonic | Funktion                       | Operation                       |
|----------|--------------------------------|---------------------------------|
| SEX      | Teckenutvidga 8 bitars operand | (A,B,CCR) →<br>(D,X,Y eller SP) |

## Introduktion till Freescale MC68HCS12

27

## Ovillkorlig programflödeskontroll

| Mnemonic | Funktion                                                                                                                                      | Operation                                                                                                                                                                          |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BSR      | Anrop av subrutin.<br>PC-relativ operand                                                                                                      | SP-2 ⇒ SP<br>RetAdRL:RetAdRH ⇒<br>M <sub>(SP)</sub> :M <sub>(SP+1)</sub><br>Adress ⇒ PC                                                                                            |
| BRA      | “Hopp” till adress.<br>PC-relativ operand                                                                                                     | Adress ⇒ PC                                                                                                                                                                        |
| CALL     | Anrop av subrutin<br>Absolut operand (20 bitar)<br>Anm: Användes vid<br>programflödesändring mellan<br>olika minnesbankar (\$8000-<br>\$BFFF) | SP-2 ⇒ SP<br>RetAdRL:RetAdRH ⇒<br>M <sub>(SP)</sub> :M <sub>(SP+1)</sub><br>Subrutinadress ⇒ PC<br>SP-1 ⇒ SP<br>(PPAGE) ⇒ M <sub>(SP)</sub><br>PAGE ⇒ PPAGE<br>Subrutinadress ⇒ PC |
| JMP      | “Hopp” till adress.<br>Absolut operand                                                                                                        | Subrutinadress ⇒ PC                                                                                                                                                                |
| JSR      | Anrop av subrutin<br>Absolut operand                                                                                                          | SP-2 ⇒ SP<br>RetAdRL:RetAdRH ⇒<br>M <sub>(SP)</sub> :M <sub>(SP+1)</sub><br>Subrutinadress ⇒ PC                                                                                    |
| RTC      | Återvänd från subrutin.<br>Returadress från STACK och<br>PPAGE                                                                                | M <sub>(SP)</sub> ⇒ (PPAGE)<br>SP+1 ⇒ SP<br>M <sub>(SP)</sub> :M <sub>(SP+1)</sub> ⇒ PC <sub>H</sub> :PC <sub>L</sub><br>SP+2 ⇒ SP                                                 |
| RTS      | Återvänd från subrutin.<br>Returadress från STACK                                                                                             | M <sub>(SP)</sub> :M <sub>(SP+1)</sub> ⇒ PC <sub>H</sub> :PC <sub>L</sub><br>SP+2 ⇒ SP                                                                                             |

## Introduktion till Freescale MC68HCS12

28

## Instruktioner för addition

| Mnemonic | Funktion                                   | Operation                   |
|----------|--------------------------------------------|-----------------------------|
| ABA      | Addera B till A                            | $(A)+(B) \rightarrow A$     |
| ABX      | Addera B till X<br>anm: Ekv. med LEAX B, X | $(X)+(B) \rightarrow X$     |
| ABY      | Addera B till Y<br>anm: Ekv. med LEAY B, Y | $(Y)+(B) \rightarrow Y$     |
| ADCA     | Addition med carry till A                  | $(A)+(M)+C \rightarrow A$   |
| ADCB     | Addition med carry till B                  | $(B)+(M)+C \rightarrow B$   |
| ADDA     | Addition till A                            | $(A)+(M) \rightarrow A$     |
| ADDB     | Addition till B                            | $(B)+(M) \rightarrow B$     |
| ADDD     | Addition till D (A:B)                      | $(D)+(M:M+1) \rightarrow D$ |

| Mnemonic | Funktion                                    | Operation                    |
|----------|---------------------------------------------|------------------------------|
| INC      | Incrementera i minnet                       | $(M)+\$01 \rightarrow M$     |
| INCA     | Inkrementera A                              | $(A)+\$01 \rightarrow A$     |
| INCB     | Inkrementera B                              | $(B)+\$01 \rightarrow B$     |
| INS      | Inkrementera SP<br>anm: Ekv. med LEAS 1, SP | $(SP)+\$0001 \rightarrow SP$ |
| INX      | Inkrementera X<br>anm: Ekv. med LEAX 1, X   | $(X)+\$0001 \rightarrow X$   |
| INY      | Inkrementera Y<br>anm: Ekv. med LEAY 1, Y   | $(Y)+\$0001 \rightarrow Y$   |

## Instruktioner för subtraktion

| Mnemonic | Funktion                     | Operation                   |
|----------|------------------------------|-----------------------------|
| SBA      | Subtrahera B från A          | $(A)-(B) \rightarrow A$     |
| SBCA     | Subtrahera med borrow från A | $(A)-(M)-C \rightarrow A$   |
| SBCB     | Subtrahera med borrow från B | $(B)-(M)-C \rightarrow B$   |
| SUBA     | Subtrahera från A            | $(A)-(M) \rightarrow A$     |
| SUBB     | Subtrahera från B            | $(B)-(M) \rightarrow B$     |
| SUBD     | Subtrahera från D (A:B)      | $(D)-(M:M+1) \rightarrow D$ |

| Mnemonic | Funktion                                     | Operation                    |
|----------|----------------------------------------------|------------------------------|
| DEC      | Dekrementera i minnet                        | $(M)-\$01 \rightarrow M$     |
| DECA     | Dekrementera A                               | $(A)-\$01 \rightarrow A$     |
| DECB     | Dekrementera B                               | $(B)-\$01 \rightarrow B$     |
| DES      | Dekrementera SP<br>anm: Ekv. med LEAS -1, SP | $(SP)-\$0001 \rightarrow SP$ |
| DEX      | Dekrementera X<br>anm: Ekv. med LEAX -1, X   | $(X)-\$0001 \rightarrow X$   |
| DEY      | Dekrementera Y<br>anm: Ekv. med LEAY -1, Y   | $(Y)-\$0001 \rightarrow Y$   |

## Instruktioner för logikoperationer

| Mnemonic | Funktion                                      | Operation                           |
|----------|-----------------------------------------------|-------------------------------------|
| ANDA     | Bitvis "och" A med minnesinnehåll             | $(A) \bullet (M) \Rightarrow A$     |
| ANDB     | Bitvis "och" A med minnesinnehåll             | $(B) \bullet (M) \Rightarrow B$     |
| ANDCC    | Bitvis "och" CC med minnesinnehåll            | $(CCR) \bullet (M) \Rightarrow CCR$ |
| EORA     | Bitvis "exklusivt eller" A med minnesinnehåll | $(A) \oplus (M) \Rightarrow A$      |
| EORB     | Bitvis "exklusivt eller" B med minnesinnehåll | $(B) \oplus (M) \Rightarrow B$      |
| ORAA     | Bitvis "eller" A med minnesinnehåll           | $(A) + (M) \Rightarrow A$           |
| ORAB     | Bitvis "eller" B med minnesinnehåll           | $(B) + (M) \Rightarrow B$           |
| ORCC     | Bitvis "eller" CCR med minnesinnehåll         | $(CCR) + (M) \Rightarrow CCR$       |

EXEMPEL: Nollställ bit 7-bit 4 på adress \$3000

```
LDAB $3000
ANDB  #00001111
STAB $3000
```

EXEMPEL: Ettställ bit 7 och bit 0 på adress \$3000

```
LDAB $3000
ORAB  #10000001
STAB $3000
```

## Unära operationer

| Mnemonic | Funktion                        | Operation                |
|----------|---------------------------------|--------------------------|
| CLC      | Nollställ carryflaggan i CCR    | $0 \Rightarrow C$        |
| CLI      | Nollställ avbrottsmask i CCR    | $0 \Rightarrow I$        |
| CLR      | Nollställ minnesinnehåll        | $\$00 \Rightarrow M$     |
| CLRA     | Nollställ A                     | $\$00 \Rightarrow A$     |
| CLRB     | Nollställ B                     | $\$00 \Rightarrow B$     |
| CLV      | Nollställ overflowflaggan i CCR | $0 \Rightarrow V$        |
| COM      | Ettkomplementera minnesinnehåll | $\$FF-(M) \Rightarrow M$ |
| COMA     | Ettkomplementera A              | $\$FF-(A) \Rightarrow A$ |
| COMB     | Ettkomplementera B              | $\$FF-(B) \Rightarrow B$ |
| NEG      | Tvåkomplementera minnesinnehåll | $\$00-(M) \Rightarrow M$ |
| NEGA     | Tvåkomplementera A              | $\$00-(A) \Rightarrow A$ |
| NEGB     | Tvåkomplementera B              | $\$00-(B) \Rightarrow B$ |

EXEMPEL: Invertera bit 2 och bit 1 på adress \$3000

```
LDAB $3000
EORB #00000110
STAB $3000
```

## Logiska skiftoperationer

| Mnemonic | Funktion                      | Operation |
|----------|-------------------------------|-----------|
| LSL      | Logiskt vänsterskift i minnet |           |
| LSLA     | Logiskt vänsterskift A        |           |
| LSLB     | Logiskt vänsterskift B        |           |
| LSLD     | Logiskt vänsterskift D        |           |
| LSR      | Logiskt högerskift i minnet   |           |
| LSRA     | Logiskt högerskift A          |           |
| LSRB     | Logiskt högerskift B          |           |
| LSRD     | Logiskt högerskift D          |           |

*Exempel på användning:*

Multiplikation med 2, tal utan tecken.

Division med 2, tal utan tecken.

## Aritmetiska skiftoperationer

| Mnemonic | Funktion                                         | Operation |
|----------|--------------------------------------------------|-----------|
| ASL      | Aritmetiskt vänsterskift i minnet (ekv. med LSL) |           |
| ASLA     | Aritmetiskt vänsterskift A (ekv. med LSLA)       |           |
| ASLB     | Aritmetiskt vänsterskift B (ekv. med LSLB)       |           |
| ASLD     | Aritmetiskt vänsterskift D (ekv. med LSLD)       |           |
| ASR      | Aritmetiskt högerskift i minnet                  |           |
| ASRA     | Aritmetiskt högerskift A                         |           |
| ASRB     | Aritmetiskt högerskift B                         |           |

*Exempel på användning, högerskift:*

Division med 2, tal med tecken.

## Instruktioner för rotation (carry-skift)

| Mnemonic | Funktion                            | Operation |
|----------|-------------------------------------|-----------|
| ROL      | Rotation vänster via carry i minnet |           |
| ROLA     | Rotation vänster via carry A        |           |
| ROLB     | Rotation vänster via carry B        |           |
| ROR      | Rotation höger via carry i minnet   |           |
| RORA     | Rotation höger via carry A          |           |
| RORB     | Rotation höger via carry B          |           |

EXEMPEL: Skifta ett 32-bitars tal på adress \$3000, 1 steg åt höger

```
LSR $3000
ROR $3001
ROR $3002
ROR $3003
```

*Exempel på användning:*

Skiftoperationer på tal större än 8 bitar.

## Instruktioner för jämförelser och test

| Mnemonic | Funktion            | Operation     |
|----------|---------------------|---------------|
| CBA      | Jämför B med A      | (A)-(B)       |
| CPA      | Jämför A med minne  | (A)-(M)       |
| CMPB     | Jämför B med minne  | (B)-(M)       |
| CPD      | Jämför D med minne  | (A:B)-(M:M+1) |
| CPS      | Jämför SP med minne | (SP)-(M:M+1)  |
| CPX      | Jämför X med minne  | (X)-(M:M+1)   |
| CPY      | Jämför Y med minne  | (Y)-(M:M+1)   |

**JÄMFÖRELSE**  
Två operander  
**BINÄR** operation

| Mnemonic | Funktion             | Operation |
|----------|----------------------|-----------|
| TST      | Testa minnesinnehåll | (M)-\$00  |
| TSTA     | Testa register A     | (A)-\$00  |
| TSTB     | Testa register B     | (B)-\$00  |

**TEST**  
En operand  
**UNÄR** operation

## Villkorlig programflödeskontroll

| Mnemonic                | Funktion                       | Villkor                |
|-------------------------|--------------------------------|------------------------|
| Enkla flaggtest         |                                |                        |
| BCS                     | "Hopp" om <i>carry</i>         | C=1                    |
| BCC                     | "Hopp" om <i>ICKE carry</i>    | C=0                    |
| BEQ                     | "Hopp" om <i>zero</i>          | Z=1                    |
| BNE                     | "Hopp" om <i>ICKE zero</i>     | Z=0                    |
| BMI                     | "Hopp" om <i>negative</i>      | N=1                    |
| BPL                     | "Hopp" om <i>ICKE negative</i> | N=0                    |
| BVS                     | "Hopp" om <i>overflow</i>      | V=1                    |
| BVC                     | "Hopp" om <i>ICKE overflow</i> | V=0                    |
| Test av tal utan tecken |                                |                        |
| BHI                     | Villkor: $R > M$               | $C + Z = 0$            |
| BHS                     | Villkor: $R \geq M$            | $C = 0$                |
| BLO                     | Villkor: $R < M$               | $C = 1$                |
| BLS                     | Villkor: $R \leq M$            | $C + Z = 1$            |
| Test av tal med tecken  |                                |                        |
| BGT                     | Villkor: $R > M$               | $Z + (N \oplus V) = 0$ |
| BGE                     | Villkor: $R \geq M$            | $N \oplus V = 0$       |
| BLT                     | Villkor: $R < M$               | $N \oplus V = 1$       |
| BLE                     | Villkor: $R \leq M$            | $Z + (N \oplus V) = 1$ |

Används typiskt tillsammans med jämförelse/test instruktioner.

### EXEMPEL

```
LDAB  $3000
CMPB  $3001
BEQ   L1
....
```

## Instruktioner för räknande programslingor

| Mnemonic | Funktion                                                                                         | Villkor                                                                                               |
|----------|--------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| DREQ     | Dekrementera innehåll i register.<br>"Hoppa" om resultatet = 0.<br>(register: A,B,D,X,Y,SP)      | (register) - 1 $\Rightarrow$ register<br>om(register)=0; "hoppa";<br>annars: nästa instruktion        |
| DBNE     | Dekrementera innehåll i register.<br>"Hoppa" om resultatet $\neq$ 0.<br>(register: A,B,D,X,Y,SP) | (register) - 1 $\Rightarrow$ register<br>om(register) $\neq$ 0; "hoppa";<br>annars: nästa instruktion |
| IREQ     | Inkrementera innehåll i register.<br>"Hoppa" om resultatet = 0.<br>(register: A,B,D,X,Y,SP)      | (register) + 1 $\Rightarrow$ register<br>om(register)=0; "hoppa";<br>annars: nästa instruktion        |
| IBNE     | Inkrementera innehåll i register.<br>"Hoppa" om resultatet $\neq$ 0.<br>(register: A,B,D,X,Y,SP) | (register) + 1 $\Rightarrow$ register<br>om(register) $\neq$ 0; "hoppa";<br>annars: nästa instruktion |
| TREQ     | Testa innehåll i register. "Hoppa"<br>om resultatet = 0.<br>(register: A,B,D,X,Y,SP)             | om(register)=0; "hoppa";<br>annars: nästa instruktion                                                 |
| TBNE     | Testa innehåll i register. "Hoppa"<br>om resultatet $\neq$ 0.<br>(register: A,B,D,X,Y,SP)        | om(register) $\neq$ 0; "hoppa";<br>annars: nästa instruktion                                          |

Sammansatta instruktioner.

### EXEMPEL

```
DBEQ  B, L2
samma sak som
DECB
BEQ   L2
```