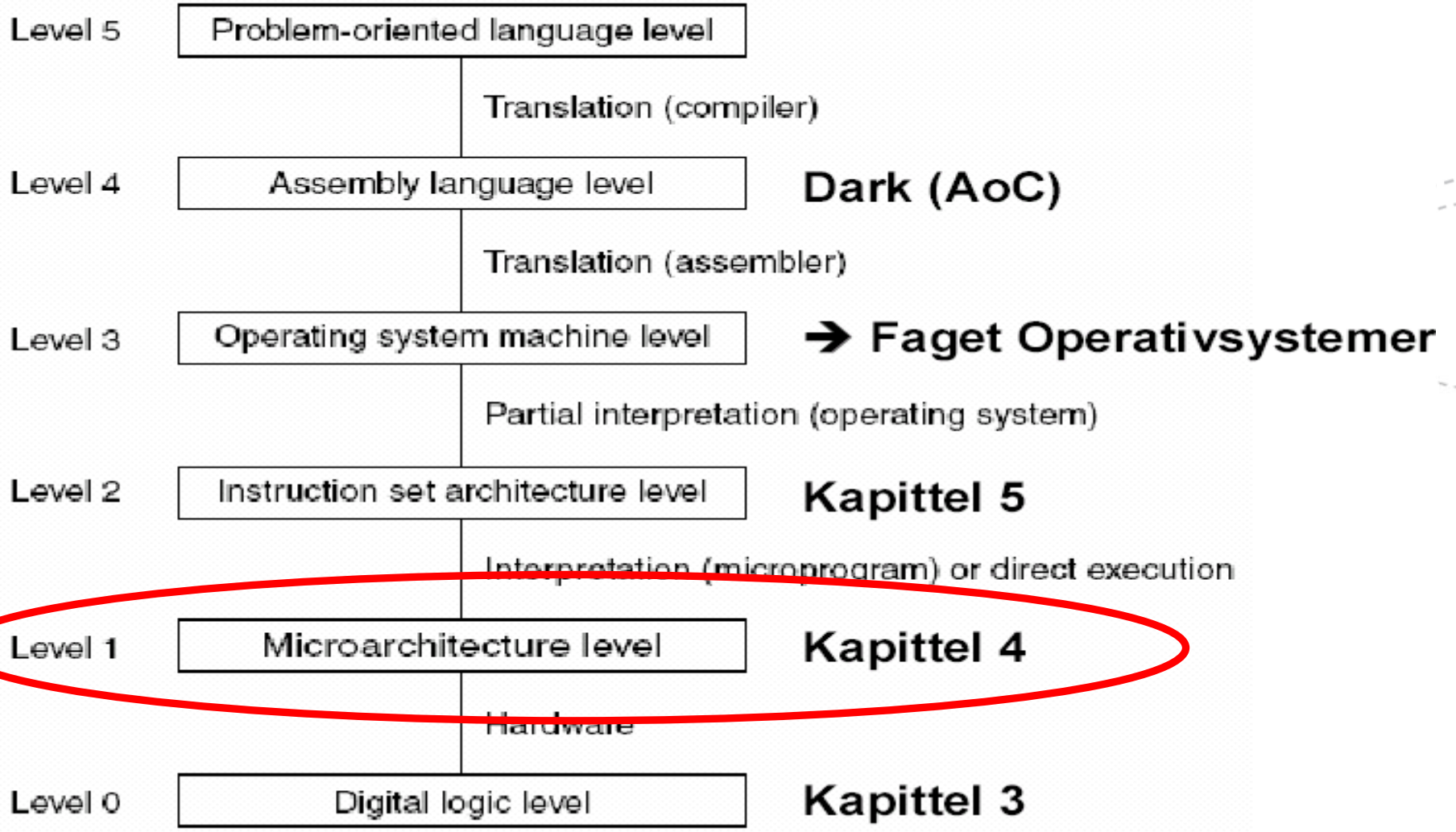
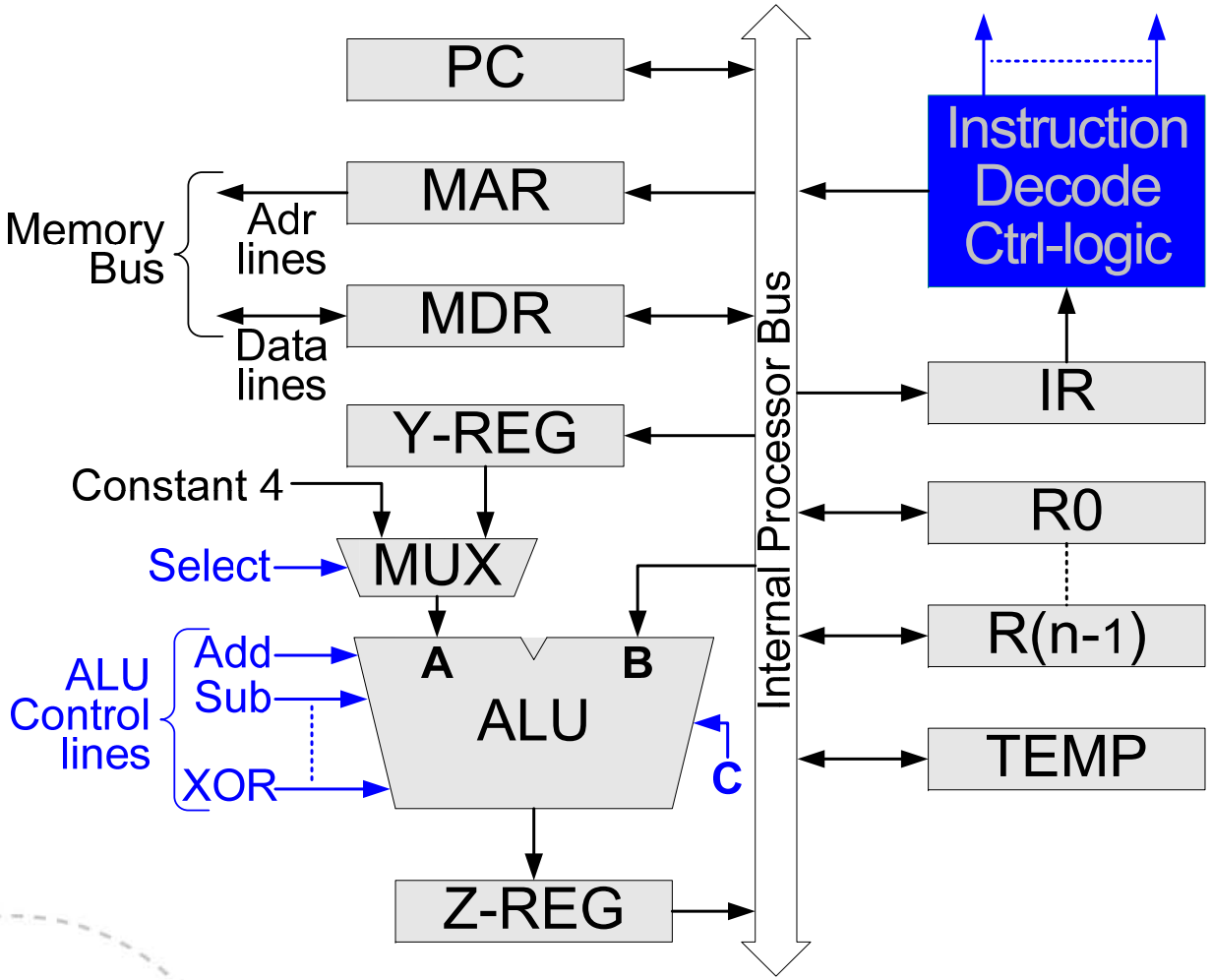
Kapittel 4: Microarchitecture level

Kapittel 4: Microarchitecture level



Kva er og Kva gjer

- Realisera Instruction Level Architecture (ISA)



NTNU
Innovation and Creativity

Nivå 2: Instruksjonssetarkitektur (ISA)

- **Instruksjonssettark. (ISA)**
- Første nivå tilgjengelig for (ekspert-)brukere
- Grense mellom maskinvare og programvare
- Opprinnelig det eneste nivået
- Språk: Maskinkode

Nivå 2: Instruksjonssetarkitektur (ISA)

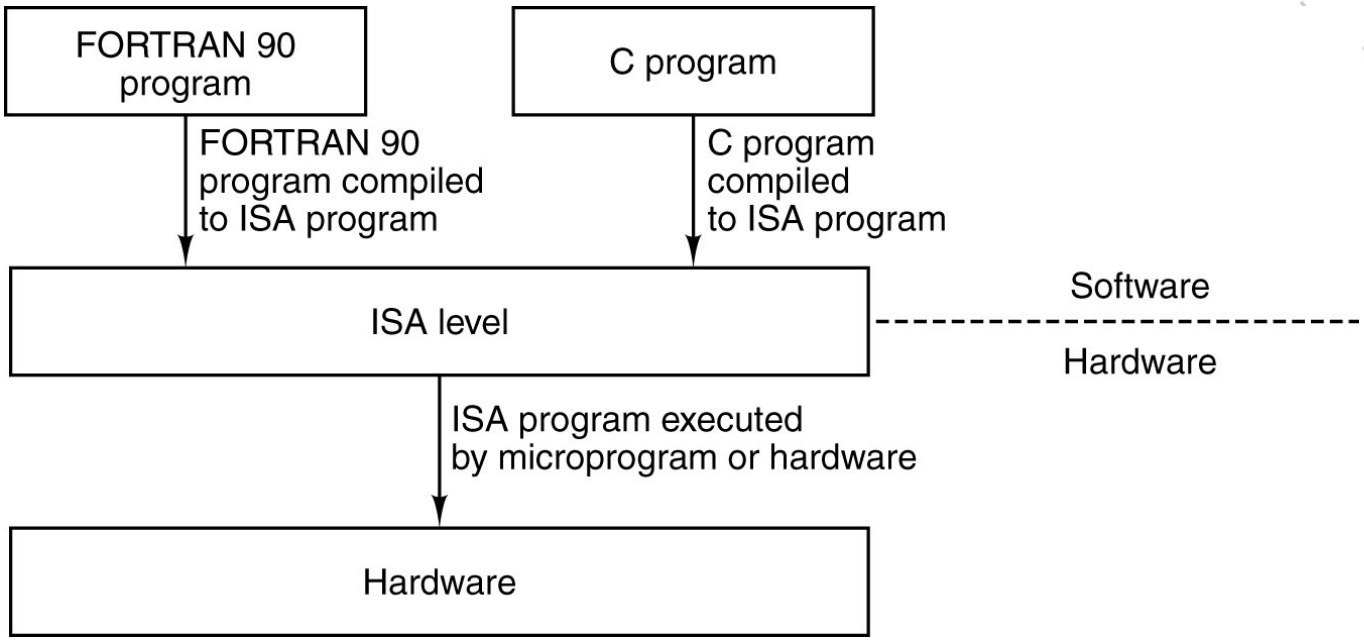
- **Instruksjonssettark. (ISA)**
- Første nivå tilgjengelig for (ekspert-)brukere
- Grense mellom maskinvare og programvare
- Opprinnelig det eneste nivået
- Språk: Maskinkode

MOV R4,A(R2):

MOV	R4	R2	124300

Nivå 2: Instruksjonssetarkitektur (ISA)

- **Instruksjonssettark. (ISA)**
- Første nivå tilgjengelig for (ekspert-)brukere
- Grense mellom maskinvare og programvare
- Opprinnelig det eneste nivået
- Språk: Maskinkode



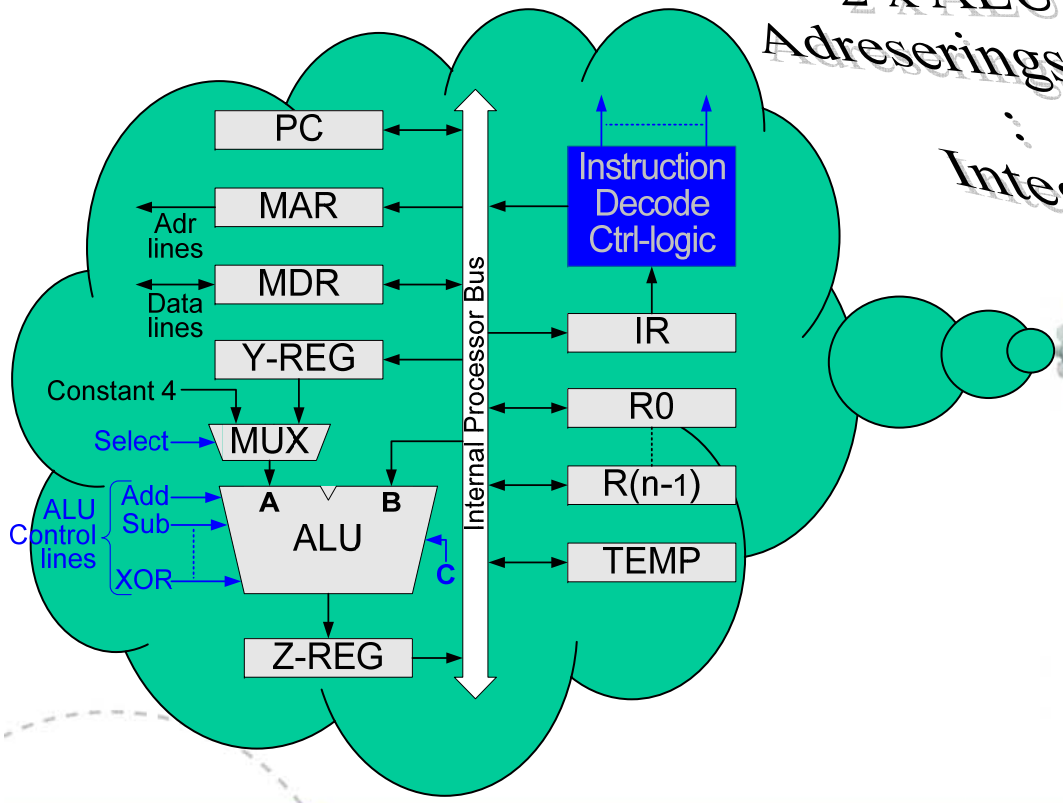
Kva er og Kva gjer

- Realisera Instruction Level Architecture (ISA)
  - Kva funksjonelle einheitar er nødvendige (generelt)
  - Kva funksjonelle einheitar er nødvendige for å oppfylle spesifikasjon
    - f.eks.:
    - Gitt av arkitektur (RISC, CISC)
    - Styreeining
    - ALU-funksjonar
    - Yting (parallellitet)
    - Instruksjonstypar
    - osv
  - Datapath
  - Kontroll av datapath

Kva er og Kva gjer

Spesifikasjon:
RISK
Superscalar
2 x ALU
Adreseringsmodi
:
Integer

Instruksjonssett:
ADD Rx Rx
LoadR mem, Rx
Store Rx, mem
:
:
AND Rx, Rx
SUB Rx, Rx

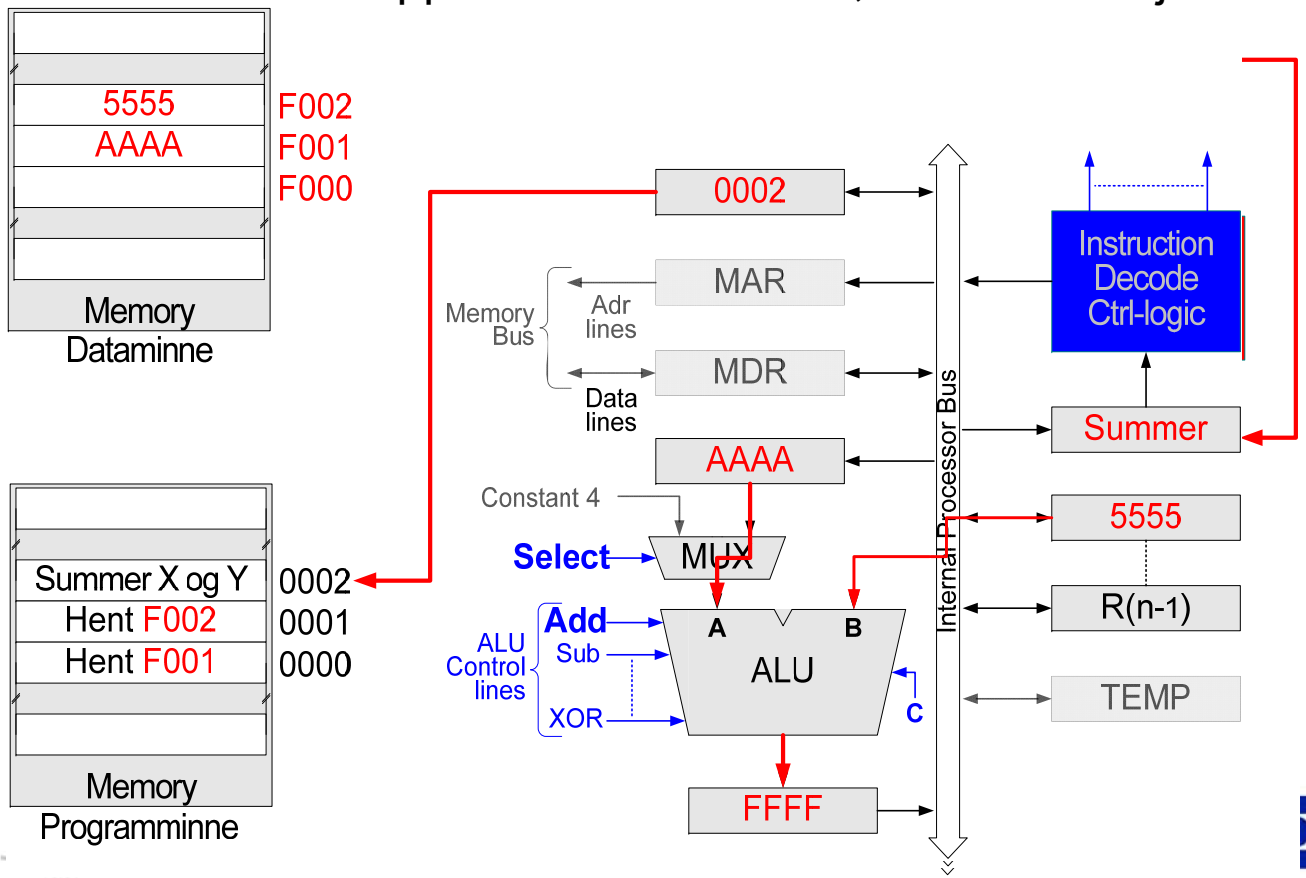


NTNU
Innovation and Creativity

Frå tidligare: instruksjons utførelse

Har henta data og lagt dei i Register 0 og Register Y

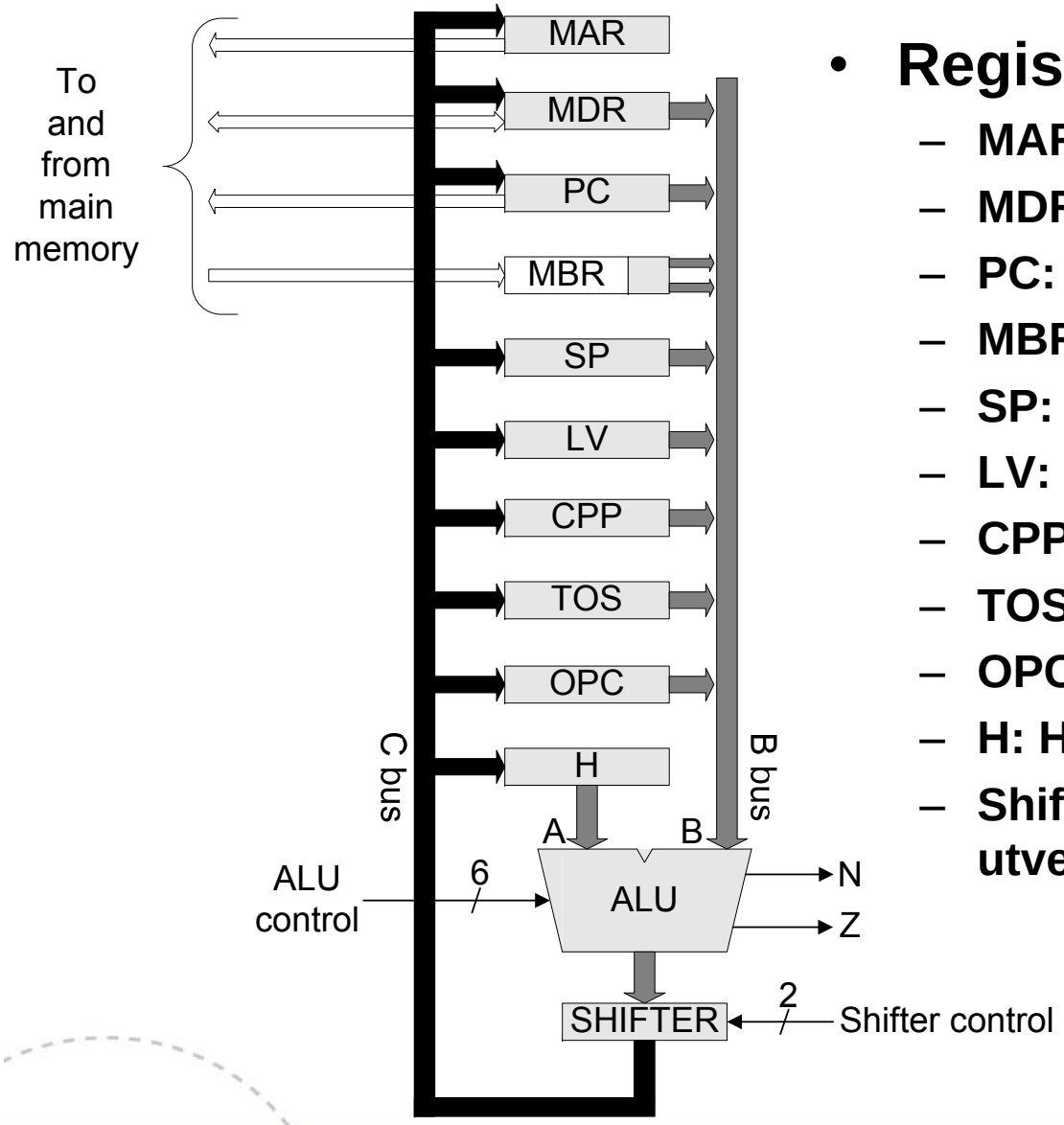
Fetch: hent neste instruksjon til Instruksjons registeret
Decode: Finn ut kva instruksjon det er
Execute: Sett opp utførandeeinheiter, utfør instruksjon



Ny Arkitektur

- **Eksempel frå boka**
- **ISA for subset av Java Virtual Machine (JVM)**
  - **Integer JVM (IJVN)**
- Kva må til for å realisere
  - Kva må vere med av utførandeeinheiter
  - Korleis må dei vere kopla saman
- Korleis kontrolere
  - Styreeinheit
  - Kontrolsignal til utførandeeinheitar

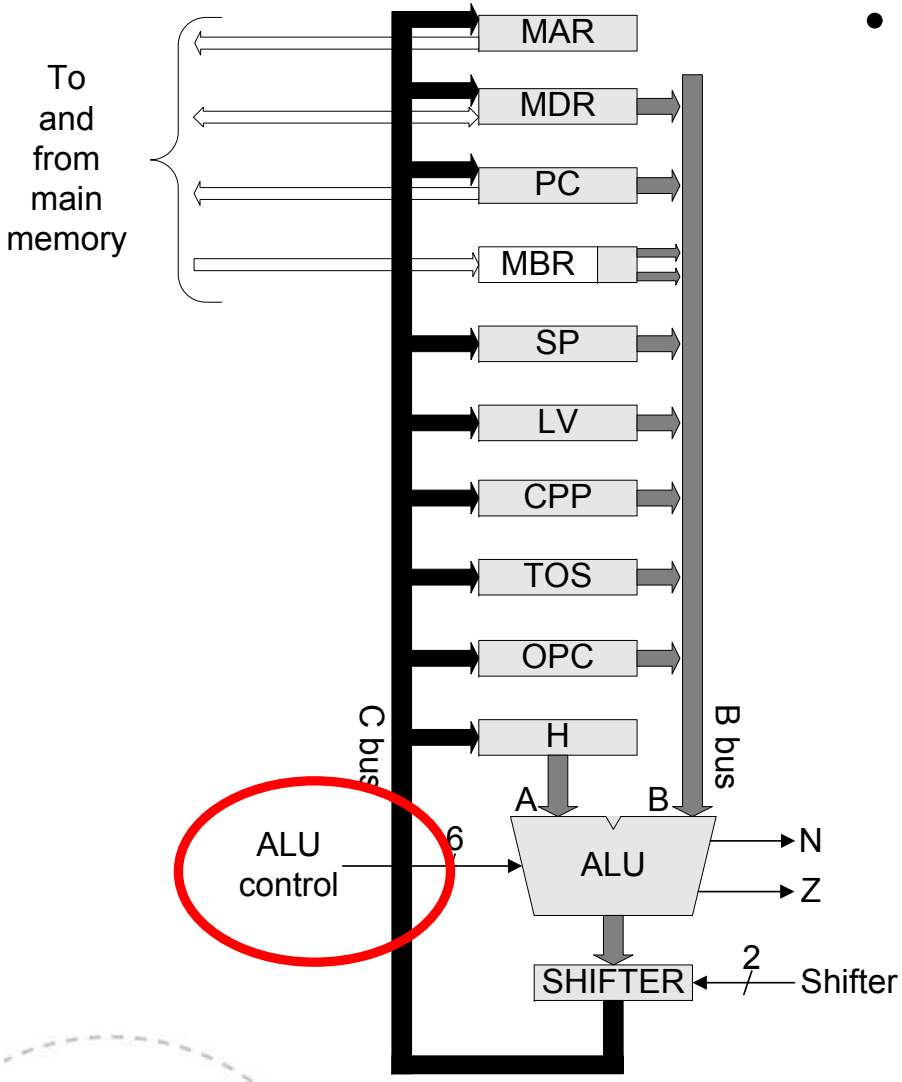
Ny Arkitektur: IJVM datapath



- **Register:**
  - **MAR: Memory Address Register**
  - **MDR: Memory Data Register**
  - **PC: Program Counter**
  - **MBR: Memory Buffer Register**
  - **SP: Stac Pointer**
  - **LV: Local variable**
  - **CPP: Constant Pool Pointer**
  - **TOS: Top of Stack**
  - **OPC: OpCode register**
  - **H: Holding register**
  - **Shifter: styrt shift register og ALU utverdi**

NTNU
Innovation and Creativity

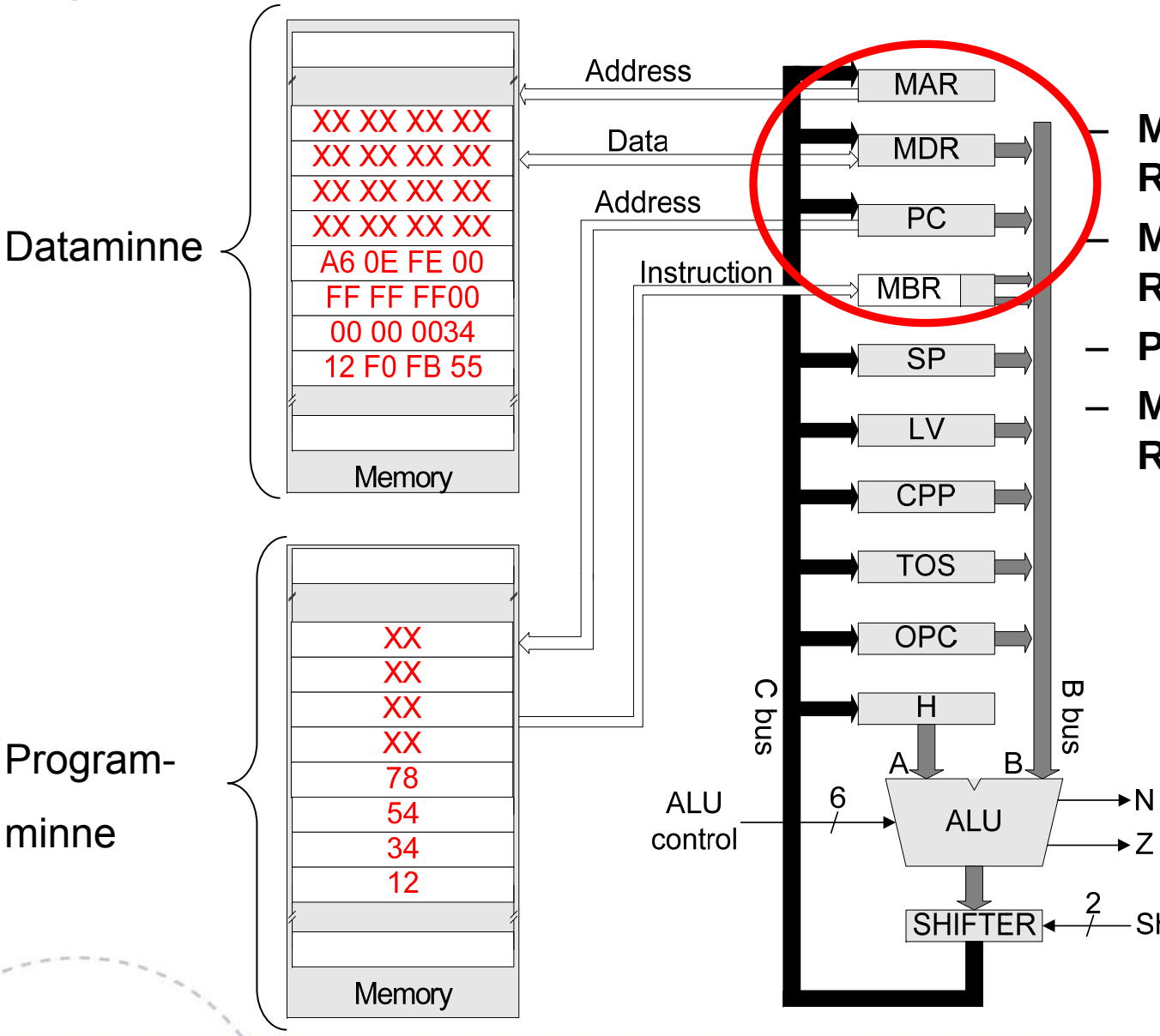
Ny Arkitektur: IJVM ALU



- **ALU : 6 kontrollinjer**

F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1

Ny Arkitektur: IJVM eksternt minne



- **MAR: Memory Address Register**
- **MDR: Memory Data Register**
- **PC: Program Counter**
- **MBR: Memory Buffer Register**

Instruksjonsett

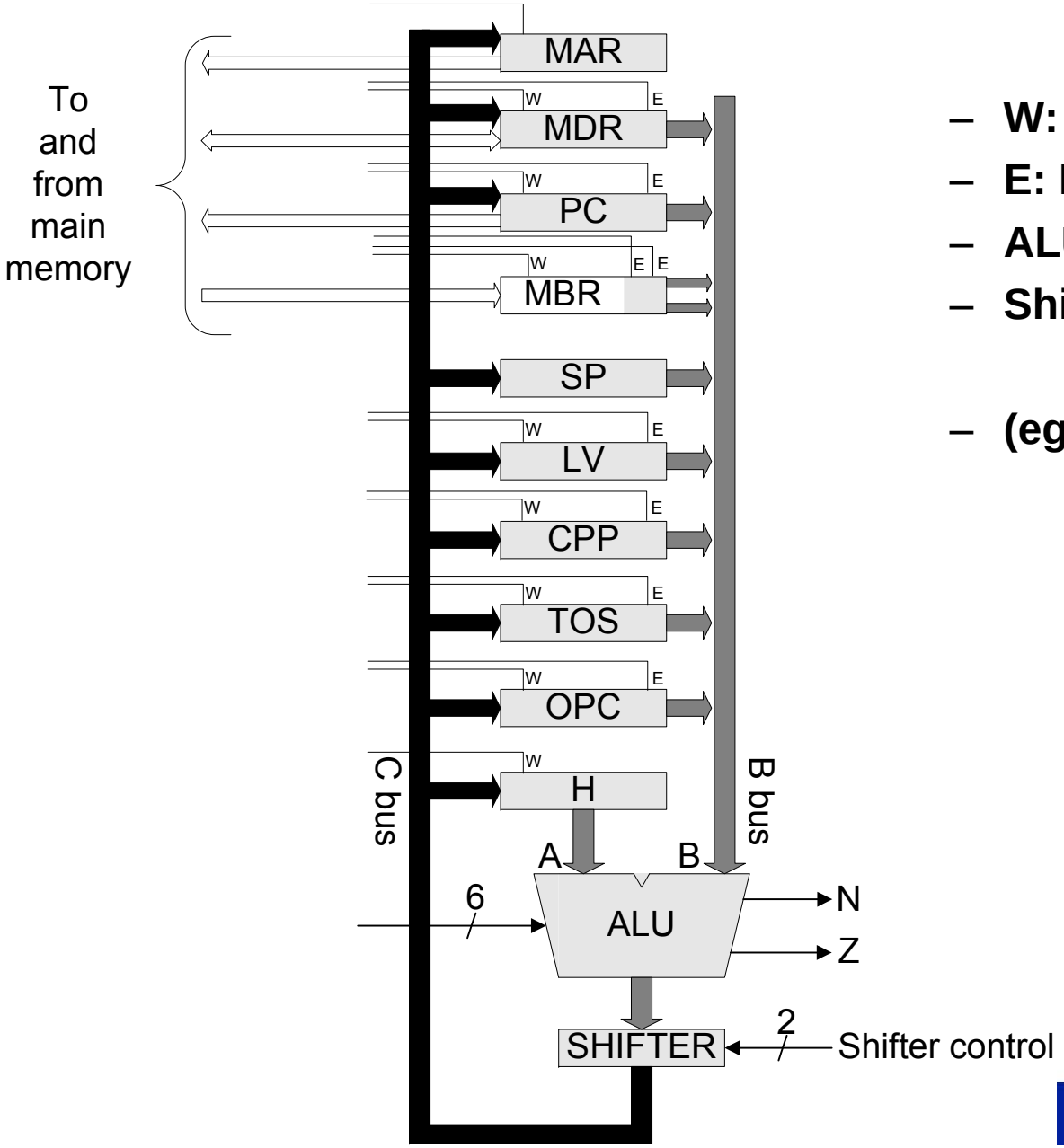
- **Kva enkle operasjonar som kan gjerast (definert)**
  - **ALU**
  - **Register verdi til buss**
  - **Last verdi frå buss til register**
  - **Minne operasjonar**
- **Kva gjer ein instruksjon?**
  - Fortel kva utførande einheitar skal gjere
- Korleis?
  - Dekoda av styreeinheit
  - Styreeinheit set opp utførande einheitar

Instruksjonsett

Hex	Mnemonic	Meaning
0x60	IADD	Pop two words from stack; push their sum
0x64	ISUB	Pop two words from stack; push their difference
0x7E	IAND	Pop two words from stack; push Boolean AND
0x80	IOR	Pop two words from stack; push Boolean OR
0x84	IINC *varnum const*	Add a constant to a local variable

- Diverse aritmetiske instruksjoner

Kontroll av utførandeeinheitar

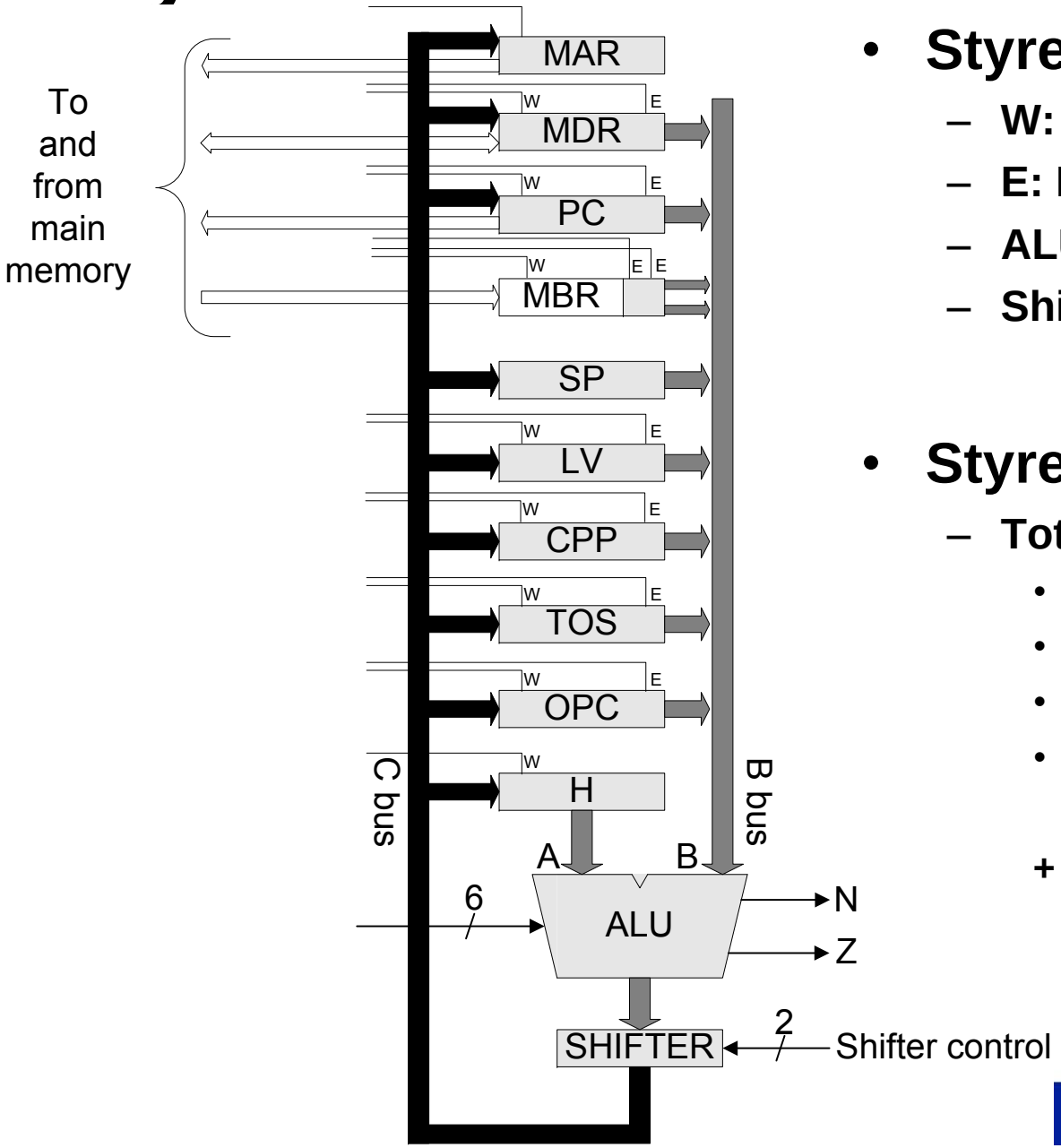


- **W: Last Register (frå C bus)**
- **E: Registerverdi til buss (til B bus)**
- **ALU: Velg funksjon**
- **Shifter: Velg funksjon**

- **(eg fekk ikkje med W og E på SP)**

NTNU
Innovation and Creativity

Styreeinheit

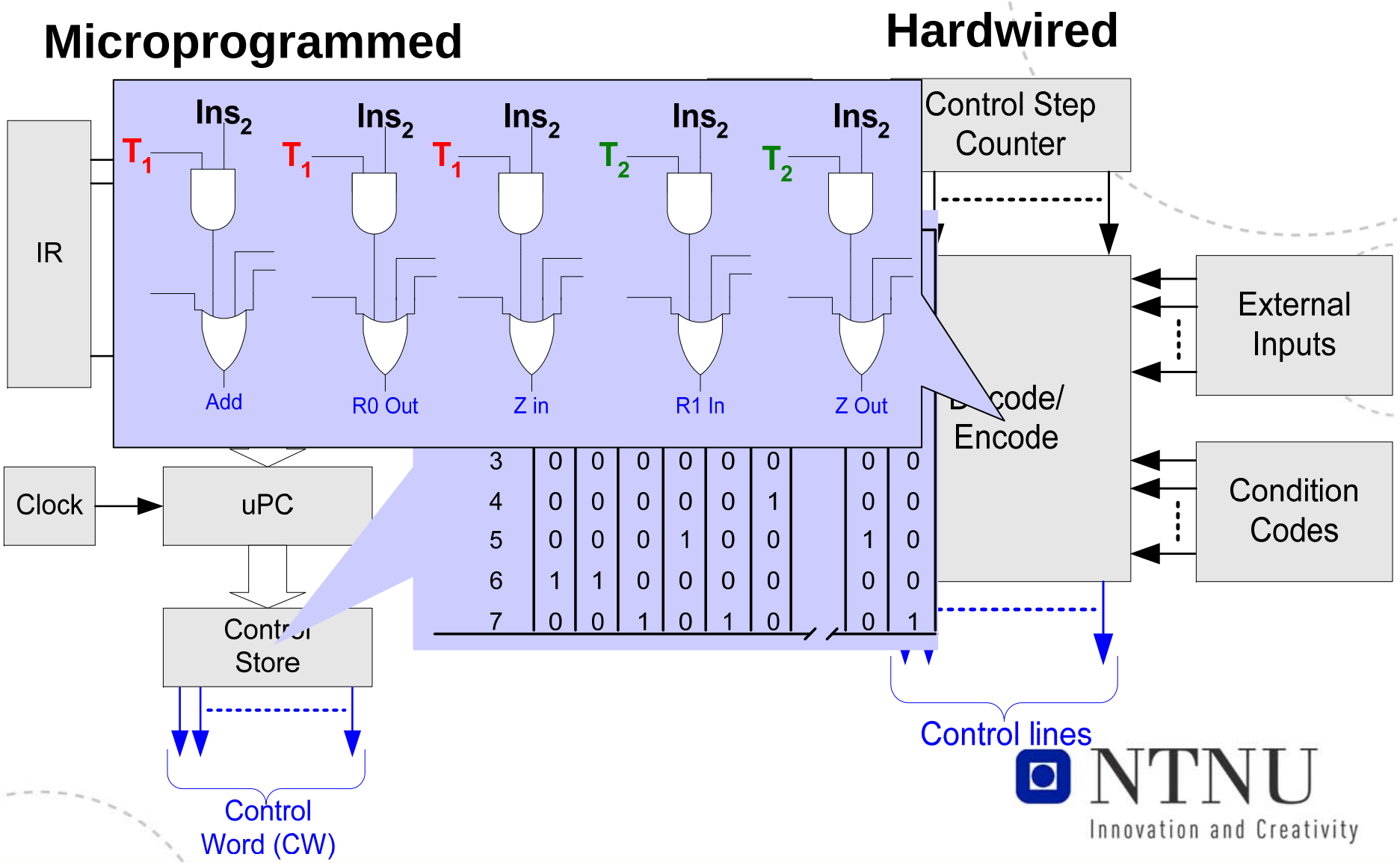


- **Styreeineit må kontrolere**
  - **W: Last Register (frå C bus)**
  - **E: Registerverdi til buss (til B bus)**
  - **ALU: Velg funksjon**
  - **Shifter: Velg funksjon**

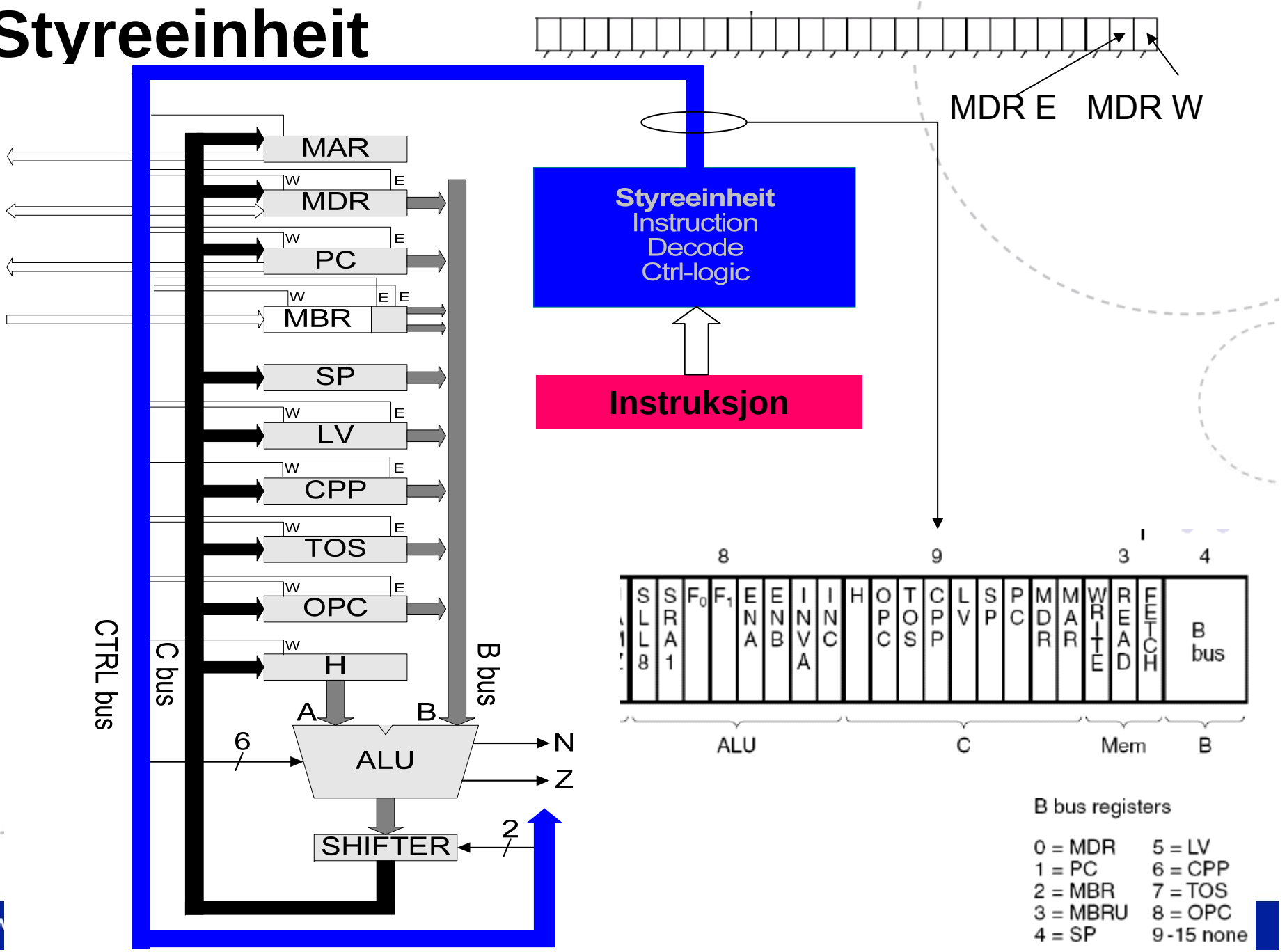
- **Styresignal:**
  - **Totalt:**
    - **6 ALU signal**
    - **2 Shift kontroll**
    - **9 Last register frå C bus (W)**
    - **9 Register til B bus (E)**

+ ekstra som me kjem til seinare

NTNU
Innovation and Creativity


Microprogrammed
Hardwired
IR
Ins2
T1
T2
Add
R0 Out
Z in
R1 In
Z Out
Control Step Counter
External Inputs
Condition Codes
Decode/ Encode
Clock
uPC
Control Store
3 0 0 0 0 0 0 0 0
4 0 0 0 0 0 1 0 0
5 0 0 0 1 0 0 1 0
6 1 1 0 0 0 0 0 0
7 0 0 1 0 1 0 0 1
Control lines
Control Word (CW)

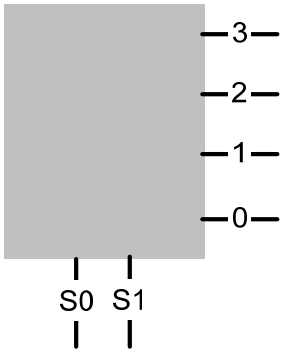
Styreeinheit



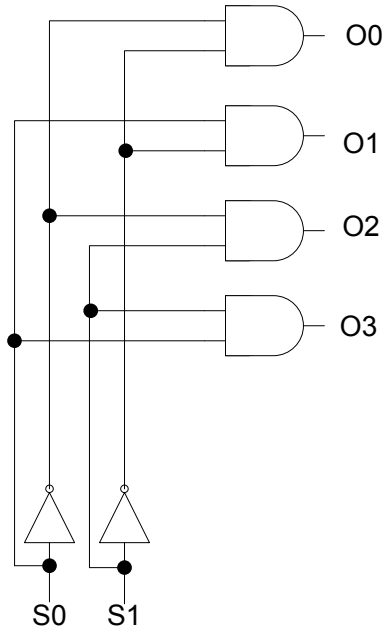
Decodar lokalt

- Kan lage kombinatoriske kretsar
  - Kretsar uten klokke eller sekvensielle eigenskapar
    - Decoders
    - Encoders
    - Multiplexers
    - Binary Adders
    - Binary Subtractors

- Eksempel kretsar: Decoder

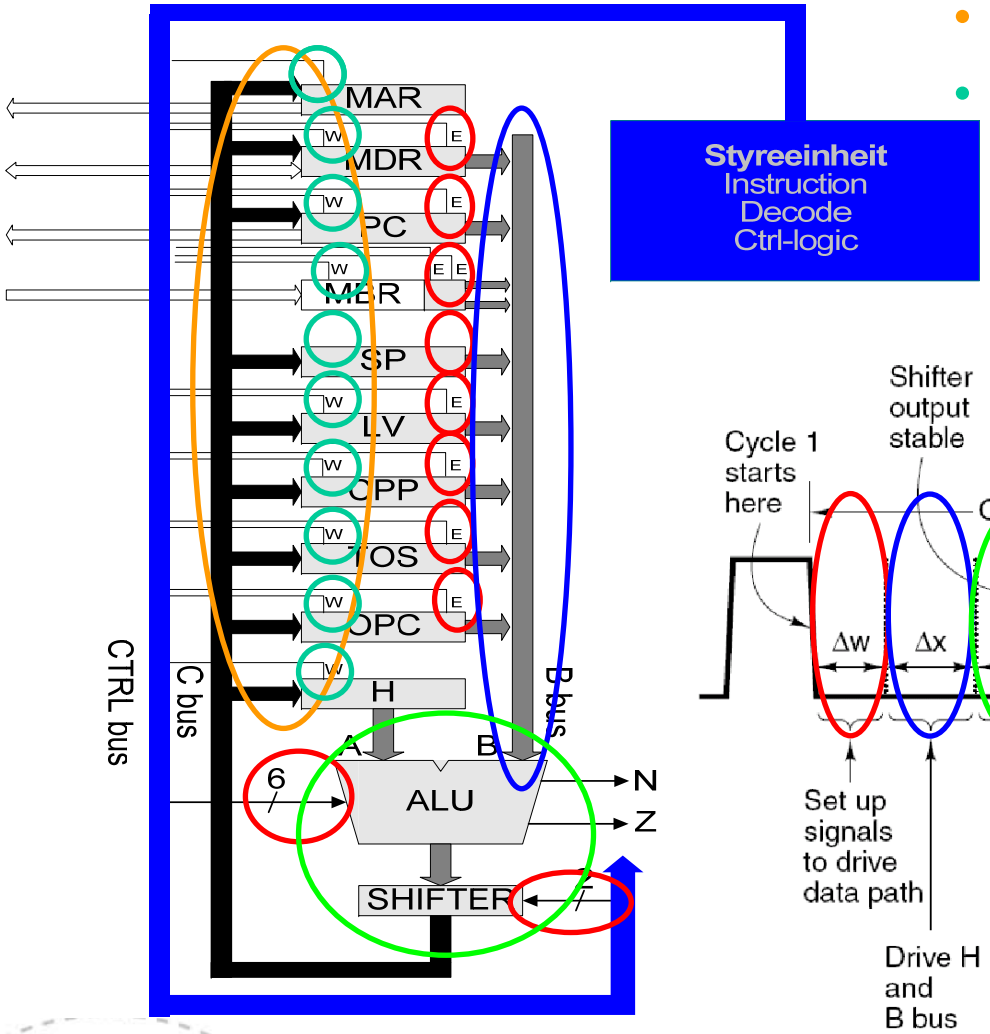


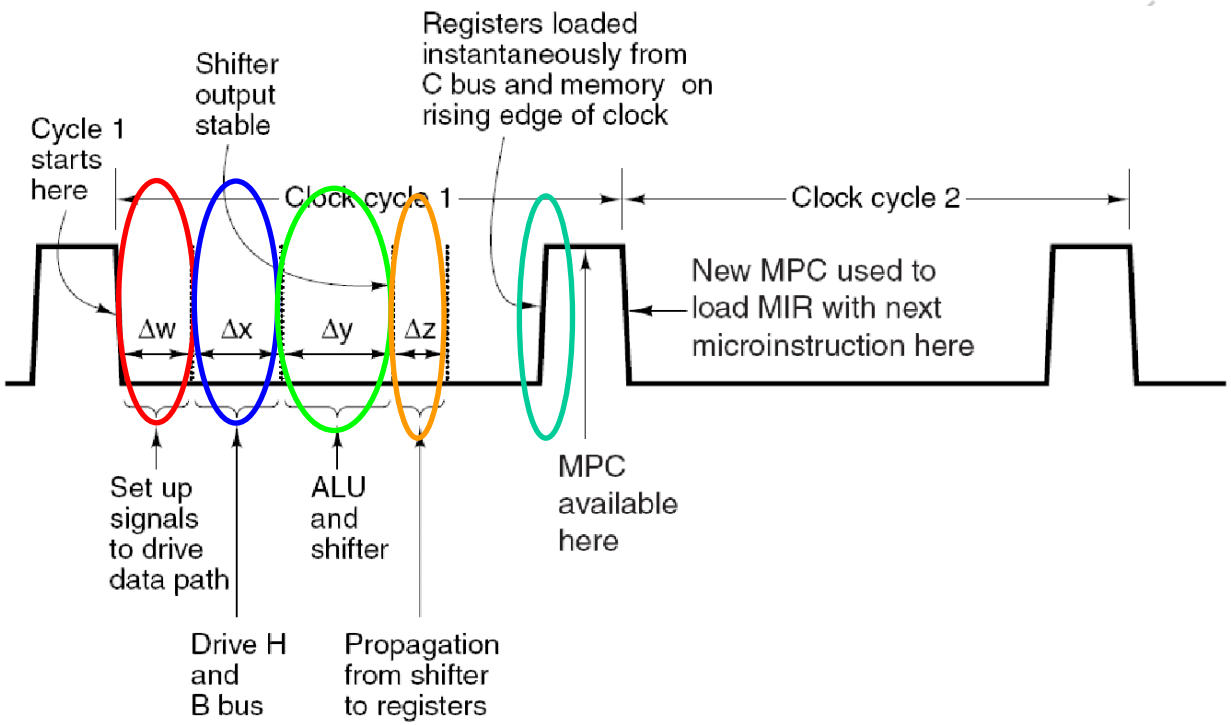
S1	S0	O3	O2	O1	O0
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Utføre instruksjon

- Set opp kontrollsignal (Δw)
- Registerverdi til B bus (Δx)
- ALU og Shifter forsinkelse (Δy)
- Forsinkelse på C bus til register (Δz)
- Last register

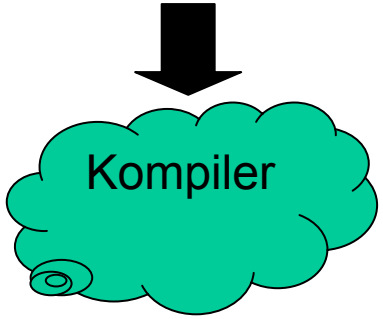




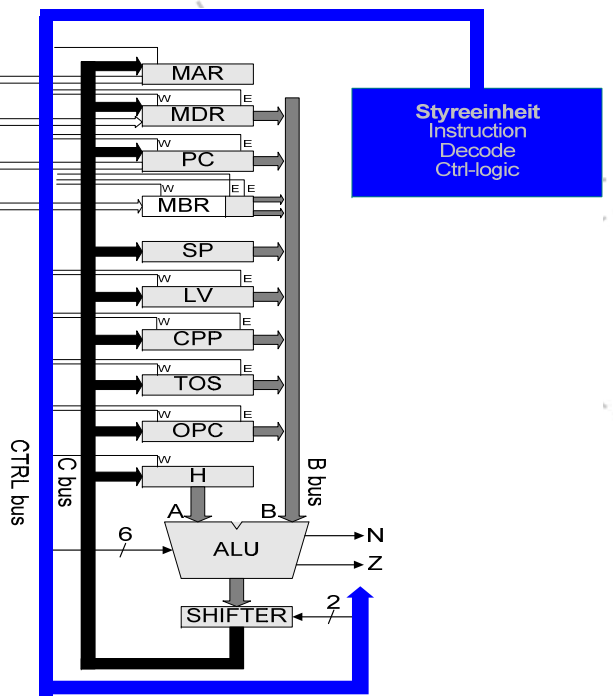
Eit lite overblikk

- Realisera Instruction Level Architecture (ISA)

Høgnivåspråk (C, C++, JAVA): **variabelC = varabelA + variabelB**;



Men egentlig me
LOAD R2, variab
LOAD R3, variab
ADD R1, R2, R3



Maskincode for + (vist som asambler): ADD R1, R2, R3

Kva skal til for å utføre ADD R1, R2, R3

Kva må styreeinheiten gjere for å få dei utførande eineitane til å:

Stille inn ALU til + opperasjon

Gi ALU tilgang på innalde i register R2 og R3

Lagre resultatet i R1

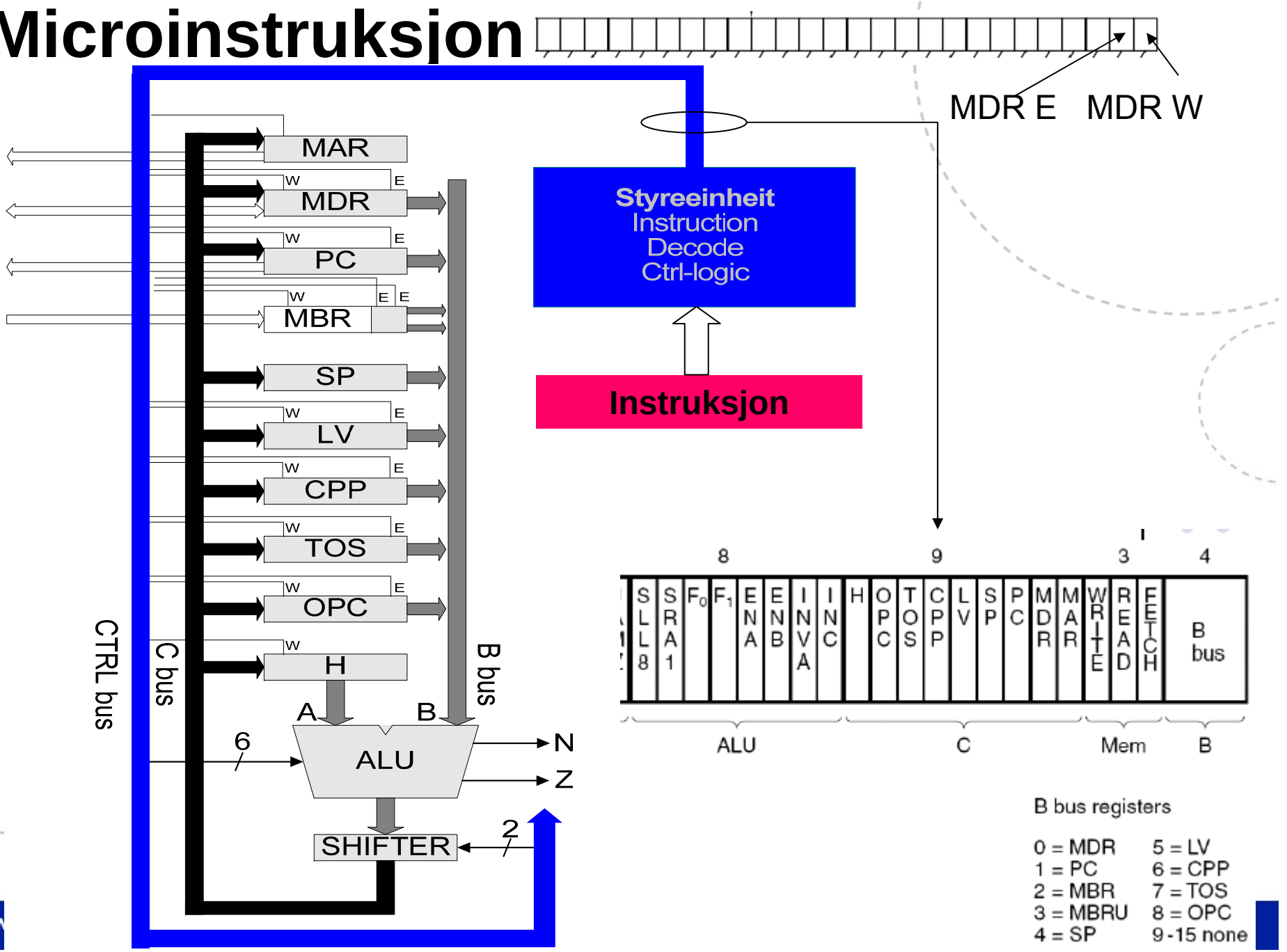
NTNU
Innovation and Creativity

For å forstå Nivå 2 Instruksjonssetarkitektur (ISA)

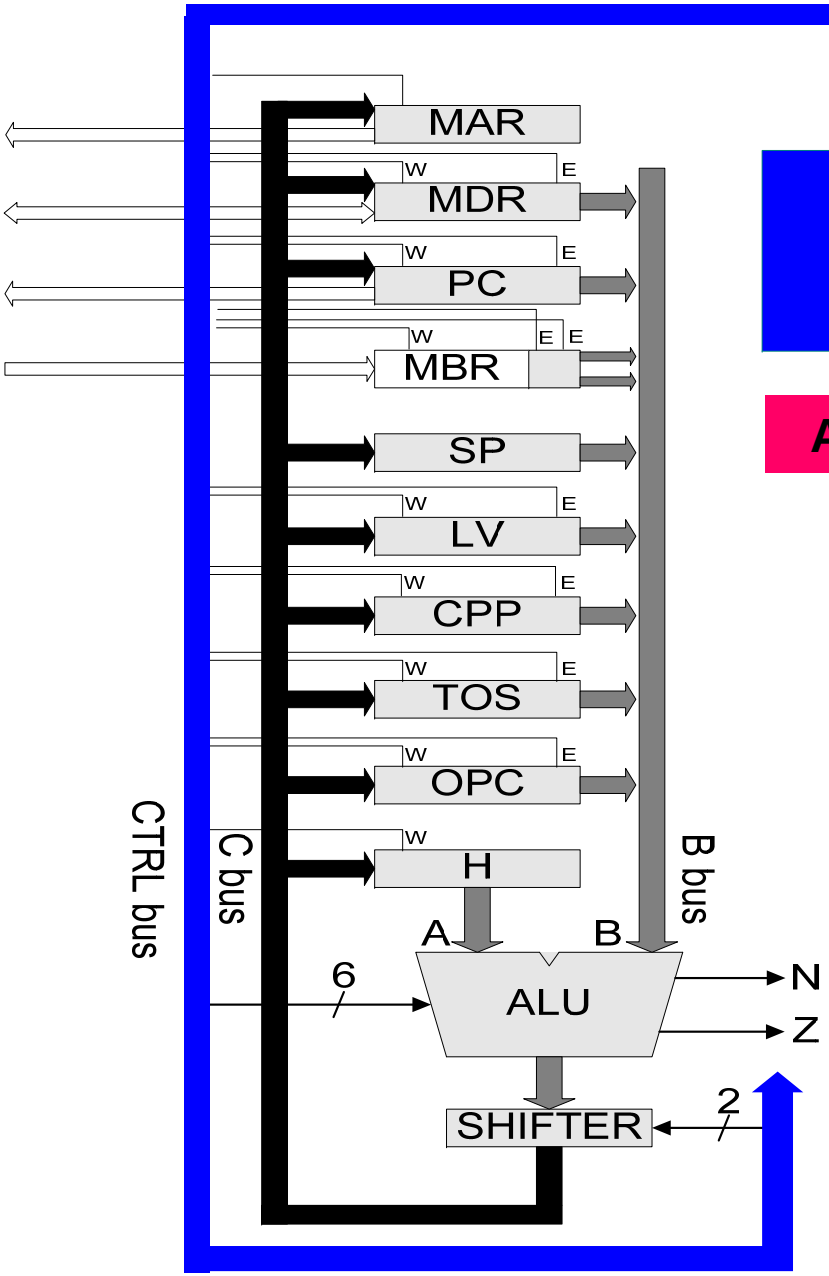
- **Instruksjonssettark. (ISA)**
  - Første nivå tilgjengelig for (ekspert-)brukere
  - Grense mellom maskinvare og programvare
  - Opprinnelig det eneste nivået
  - Språk: Maskinkode
- ISA er tilgjengeleg for programmerar
  - Viser kva ein instruksjon faktisk gjer
    - ADD R1, R2, R3
      - Resultat = R1
      - Data = R2
      - Data = R3
      - Utførest på ein klokkeperiode
      - Instruksjonen inkluderar ZERO, NEGATIV og CARRY flagg
- Nyttig for å forstå kva ISA er og for å bruke ISA-nivået
- Må kunne for å lage prosessorar

NTNU
Innovation and Creativity

Microinstruksjon



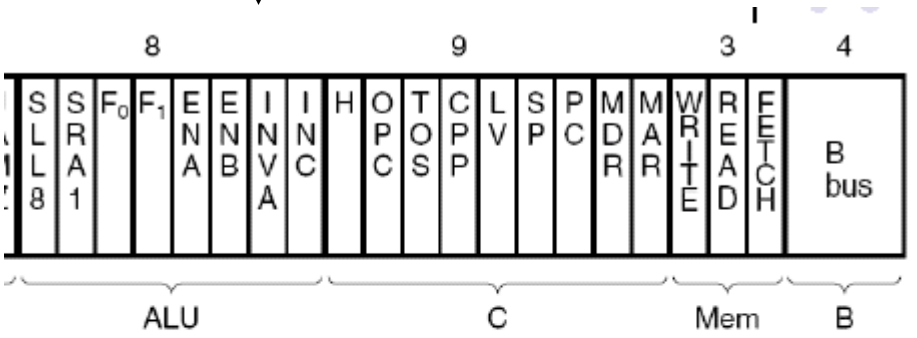
Microinstruksjonsformat



Styreeinheit
Instruction
Decode
Ctrl-logic

ADD Reg1, Reg2

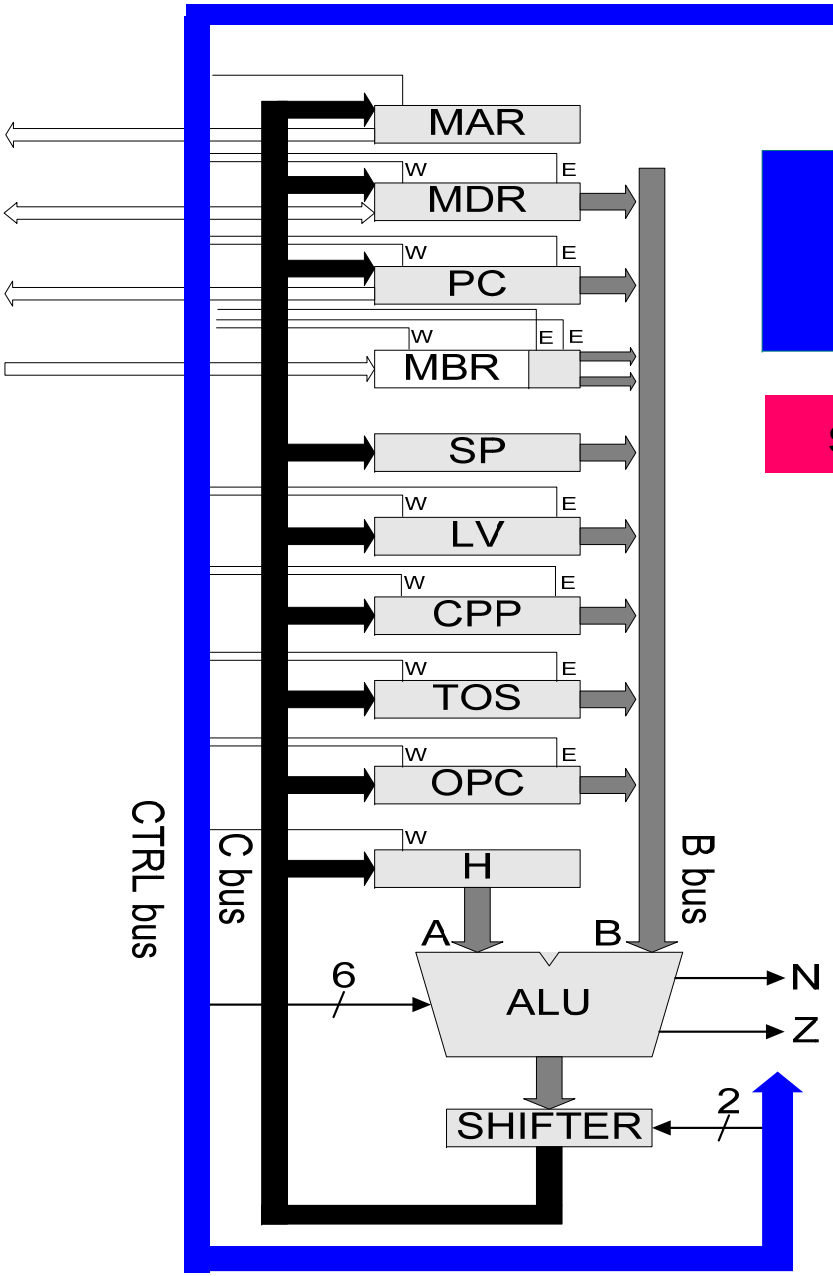
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1



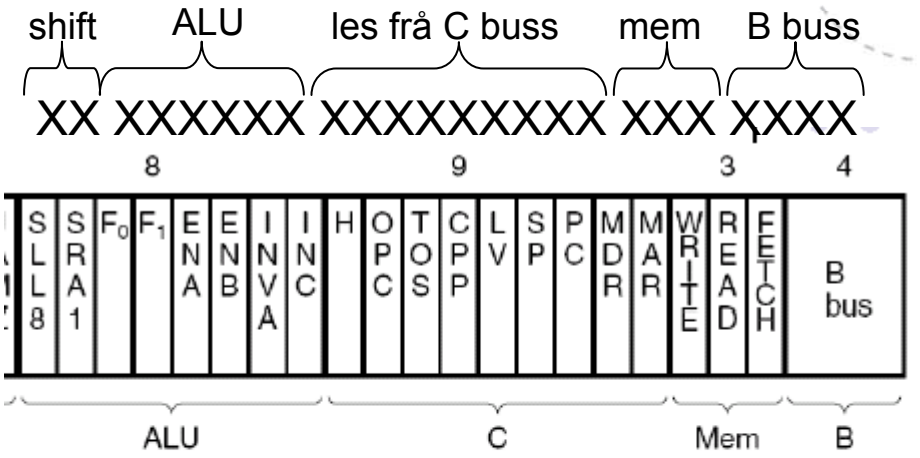
B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Eksempel: 1 (supersimpel)



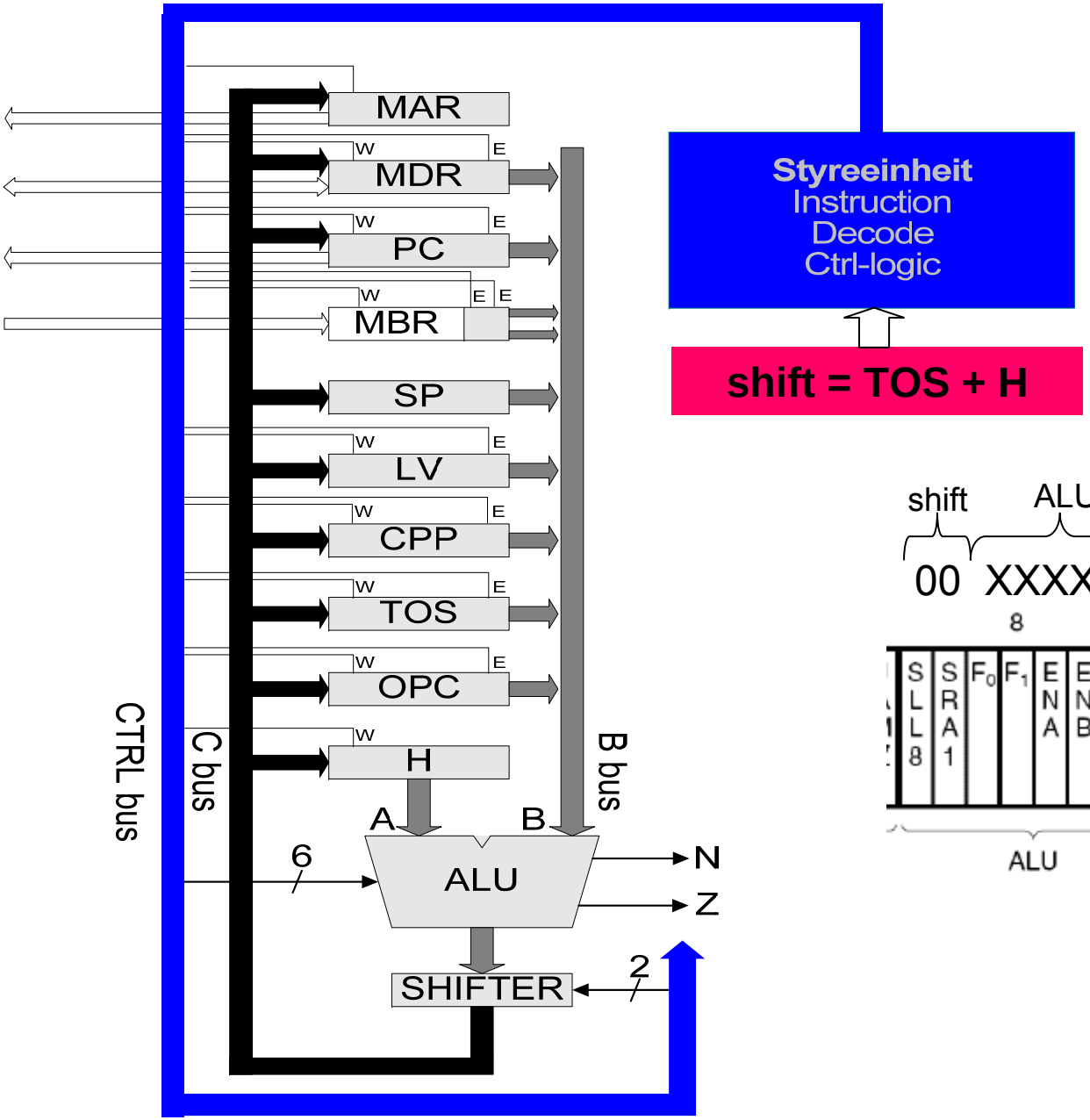
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1



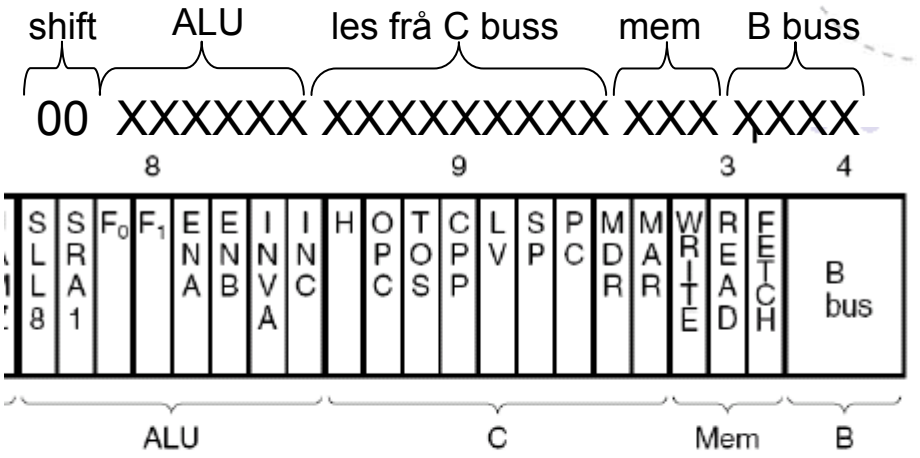
B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Eksempel: 1 (supersimpel)



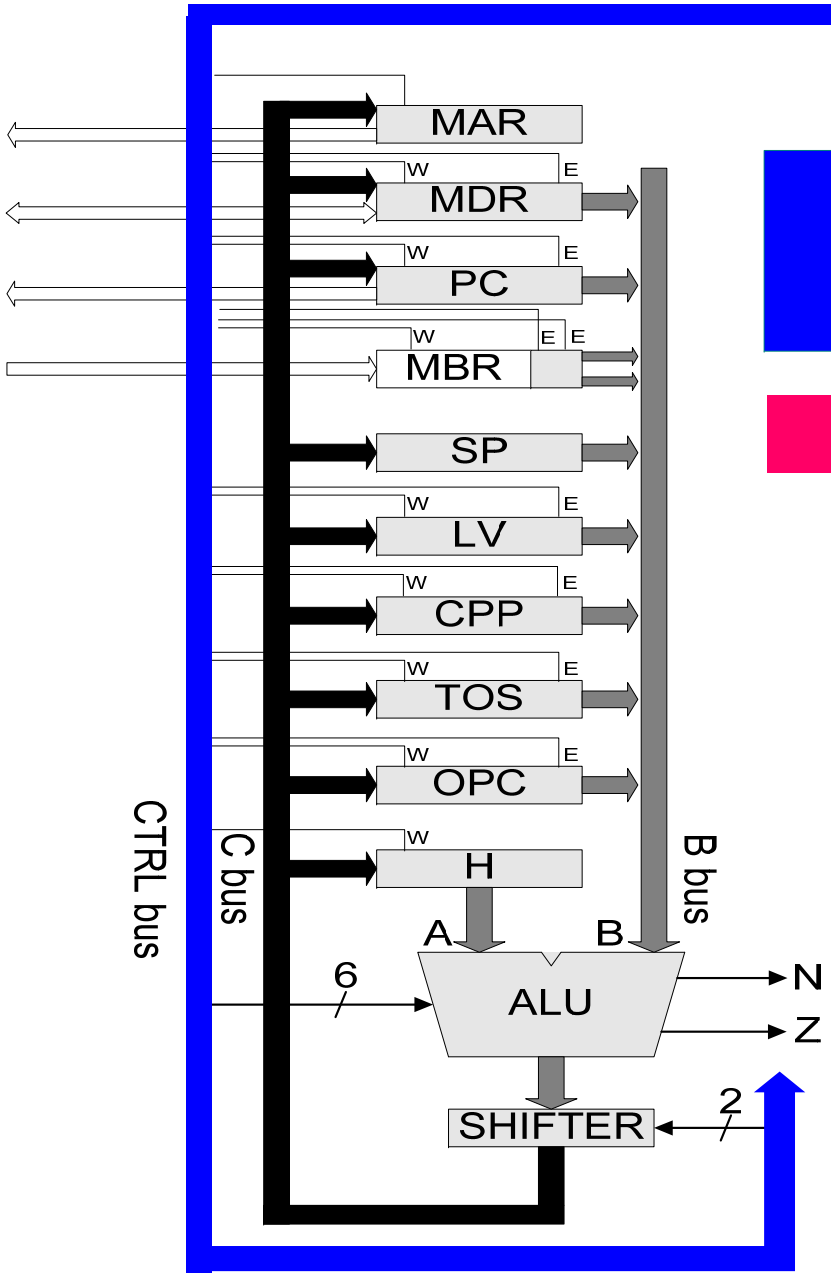
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1



B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

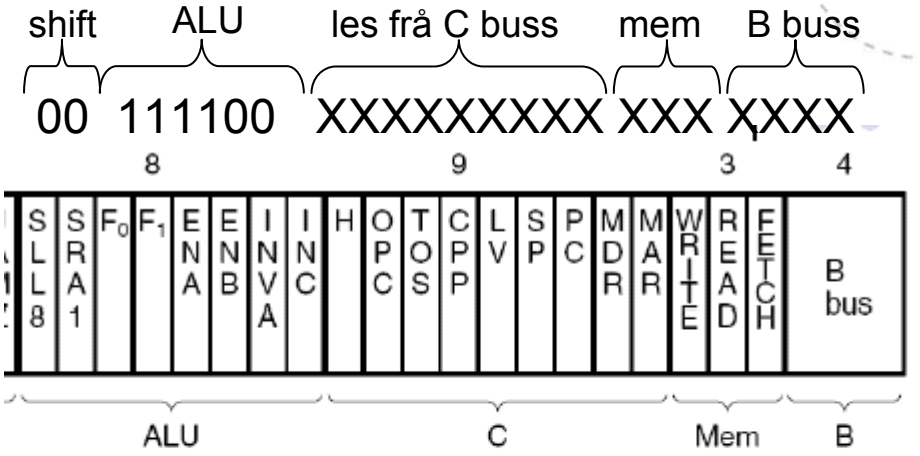
Eksempel: 1 (supersimpel)



Styreeinheit
Instruction
Decode
Ctrl-logic

shift = TOS + H

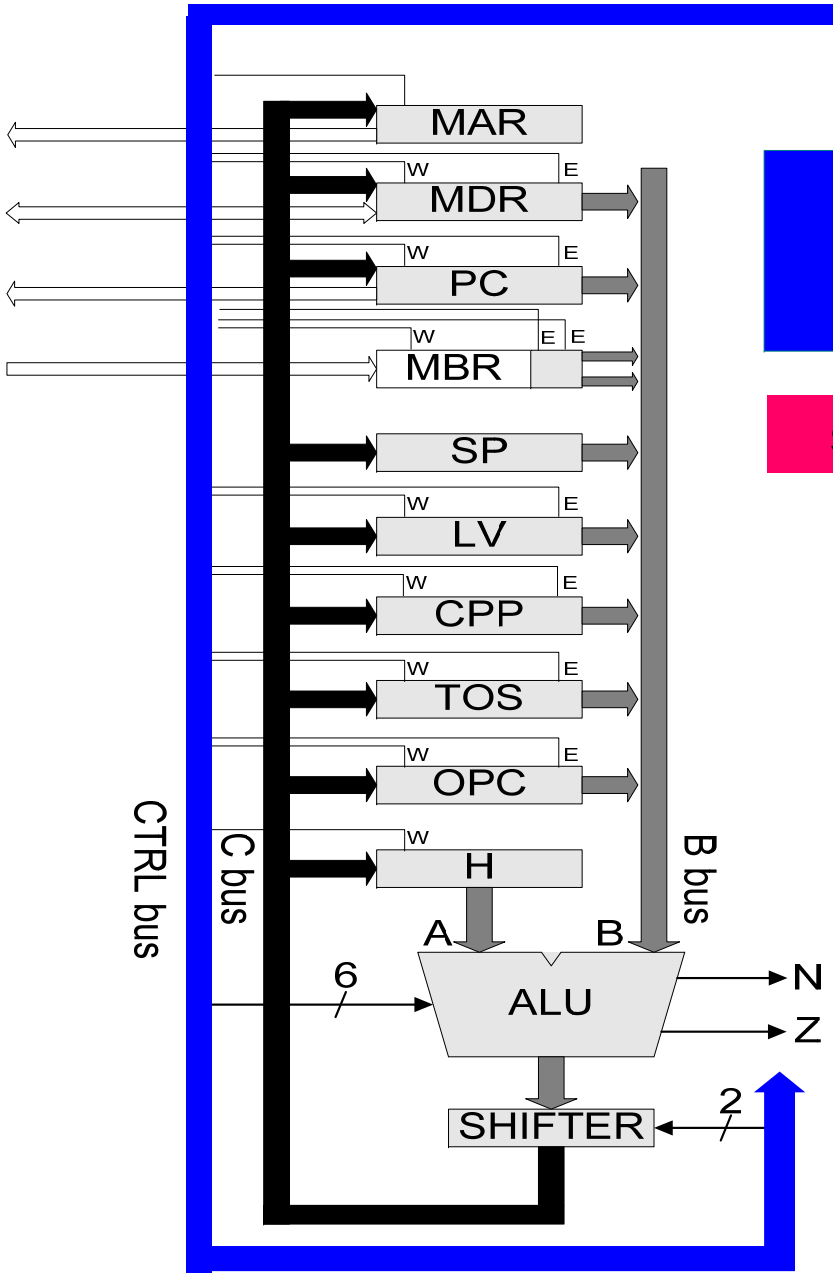
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1



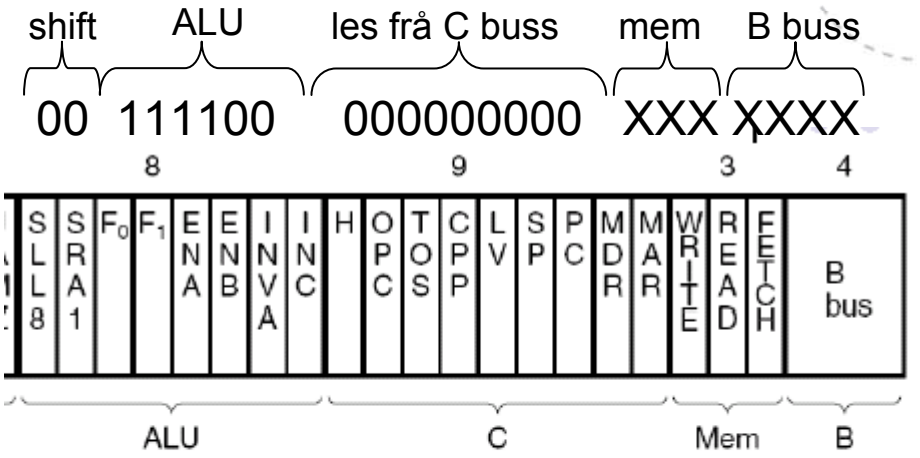
B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Eksempel: 1 (supersimpel)



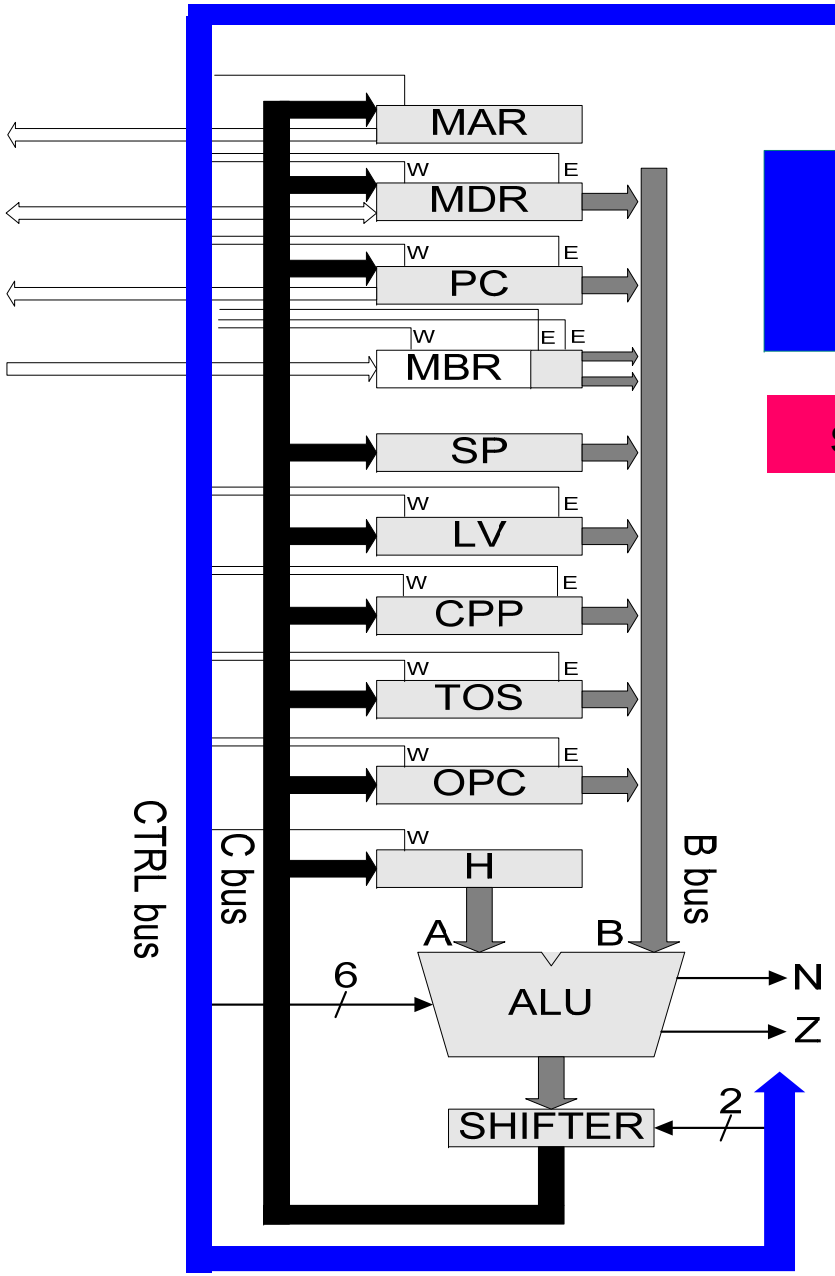
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1



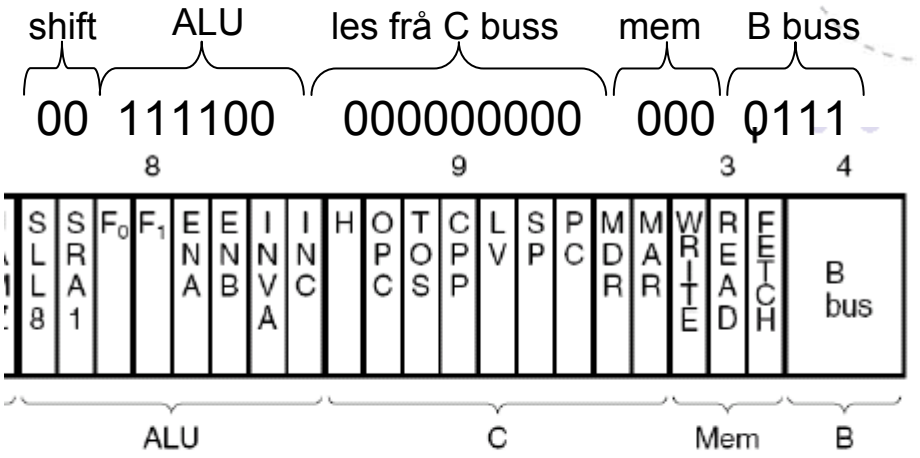
B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Eksempel: 1 (supersimpel)



F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1

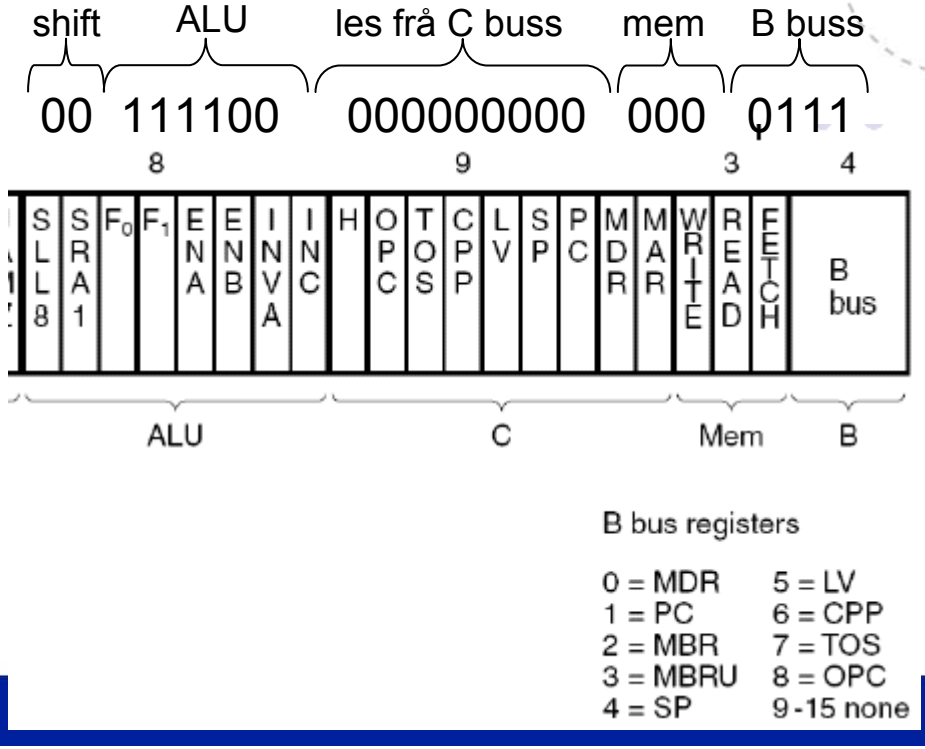


B bus registers

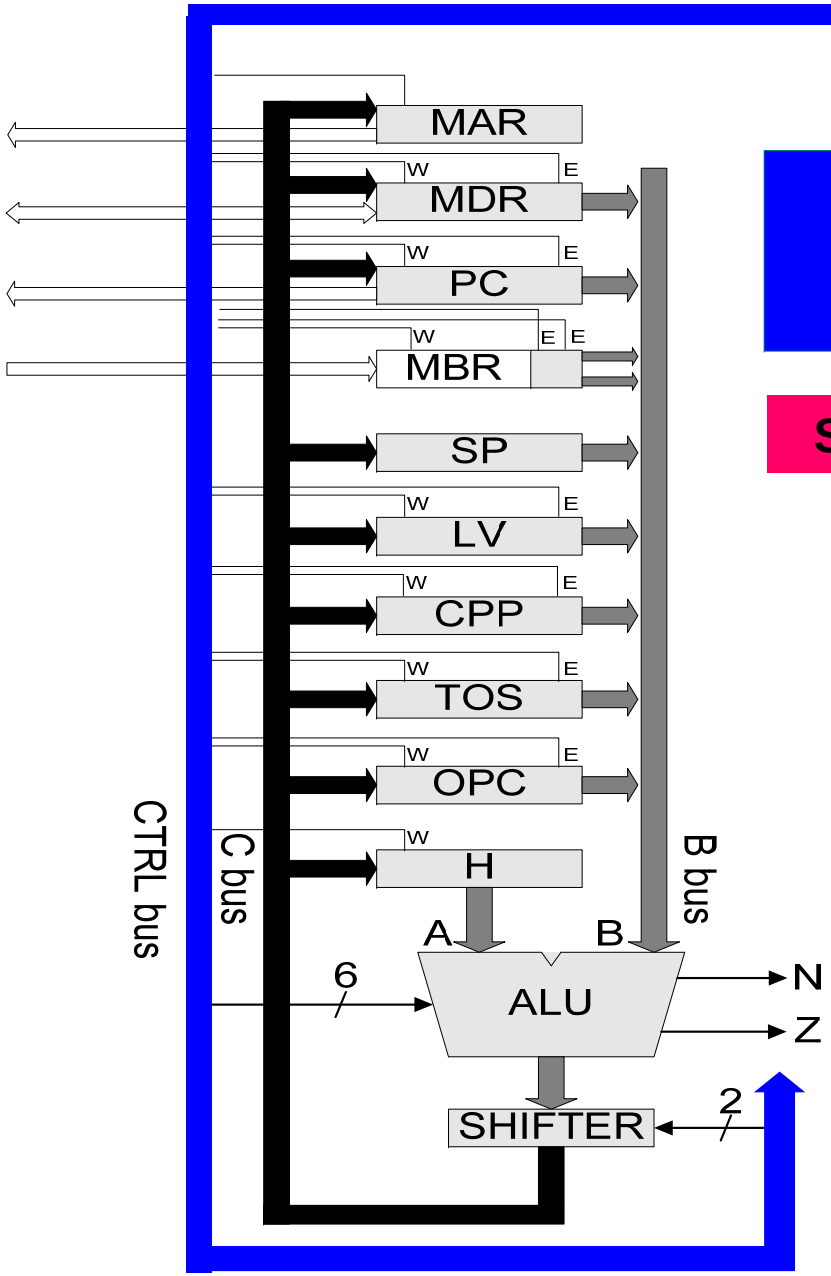
0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

har laga ein microinstruksjon

- **Instruksjon: shift = TOS + H**
- ISA er tilgjengeleg for programmerar
  - Viser kva ein instruksjon faktisk gjer på ISA nivå
    - shift = TOS + H (eksempel: **ADD RegX** (RegX er i vårt eksempel TOS))
    - Resultat = shift
    - Data = H (implicit ALU + funksjon brukar alltid H-register)
    - Data = TOS
    - Utførest på ein klokkeperiode
    - Flagg ????
- **På ISA nivå: ADD Reg X**
  - kompilator velger microinstruksjon
    - frå register på B buss f.eks:
    - shift = TOS + H (B buss = 0111 (7))
    - shift = OPC + H (B buss = 1000 (0))
    - Shift = SP + H (B buss = 0100 (4))
  - Sjult for programerar
    - brukar berre ISA nivå



Eksempel: 2 brukar B og C buss



Styreeinheit
Instruction
Decode
Ctrl-logic

SP = TOS + OPC

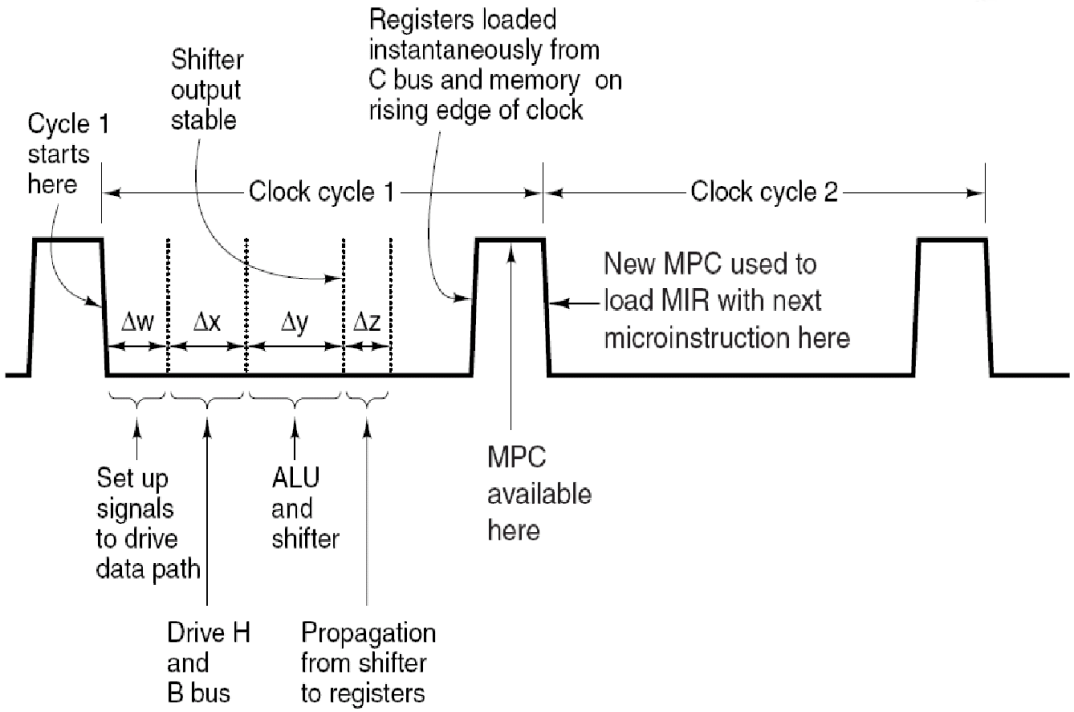
TOS eller OPC må fyrst i H-register: (velger TOS)

Instruksjon må då:

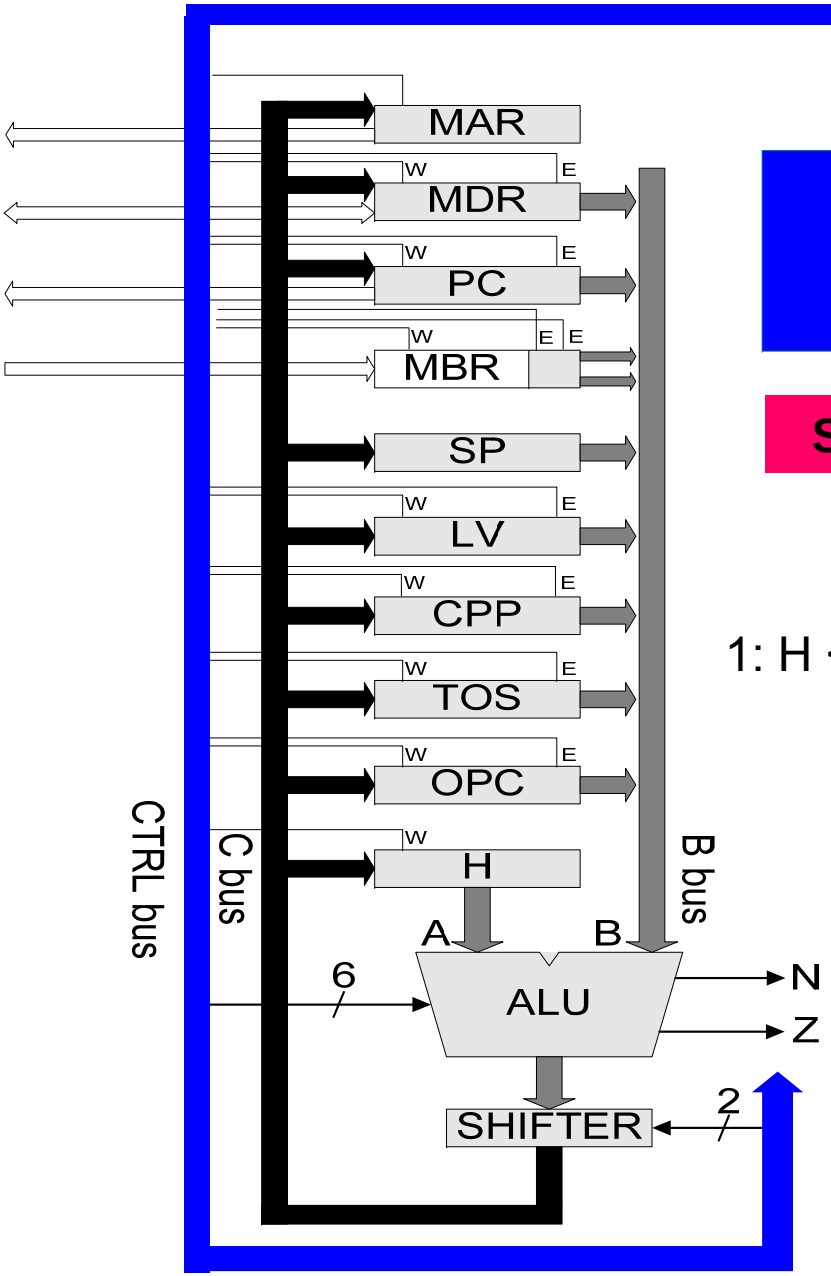
1: H <- TOS

2: H + OPC, SP <- SHIFT

To microinstruksjonar



Eksempel: 2

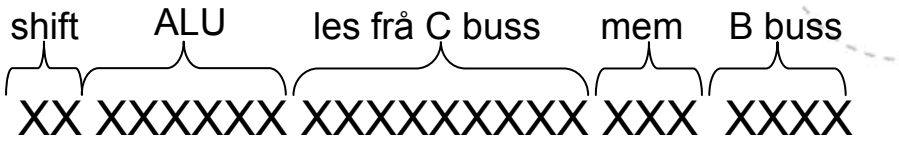


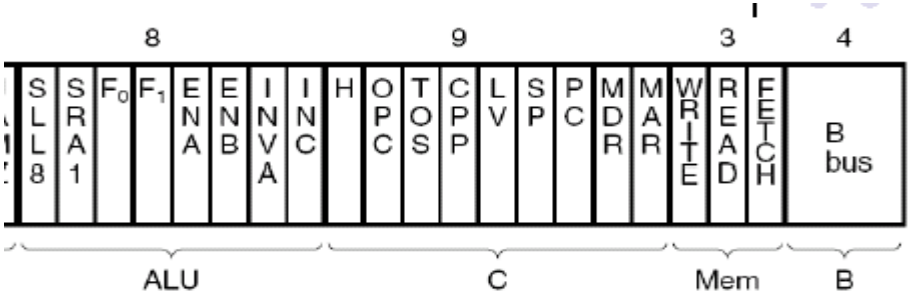
Styreeinheit
Instruction
Decode
Ctrl-logic

SP = TOS + OPC

F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1

1: H <- TOS:

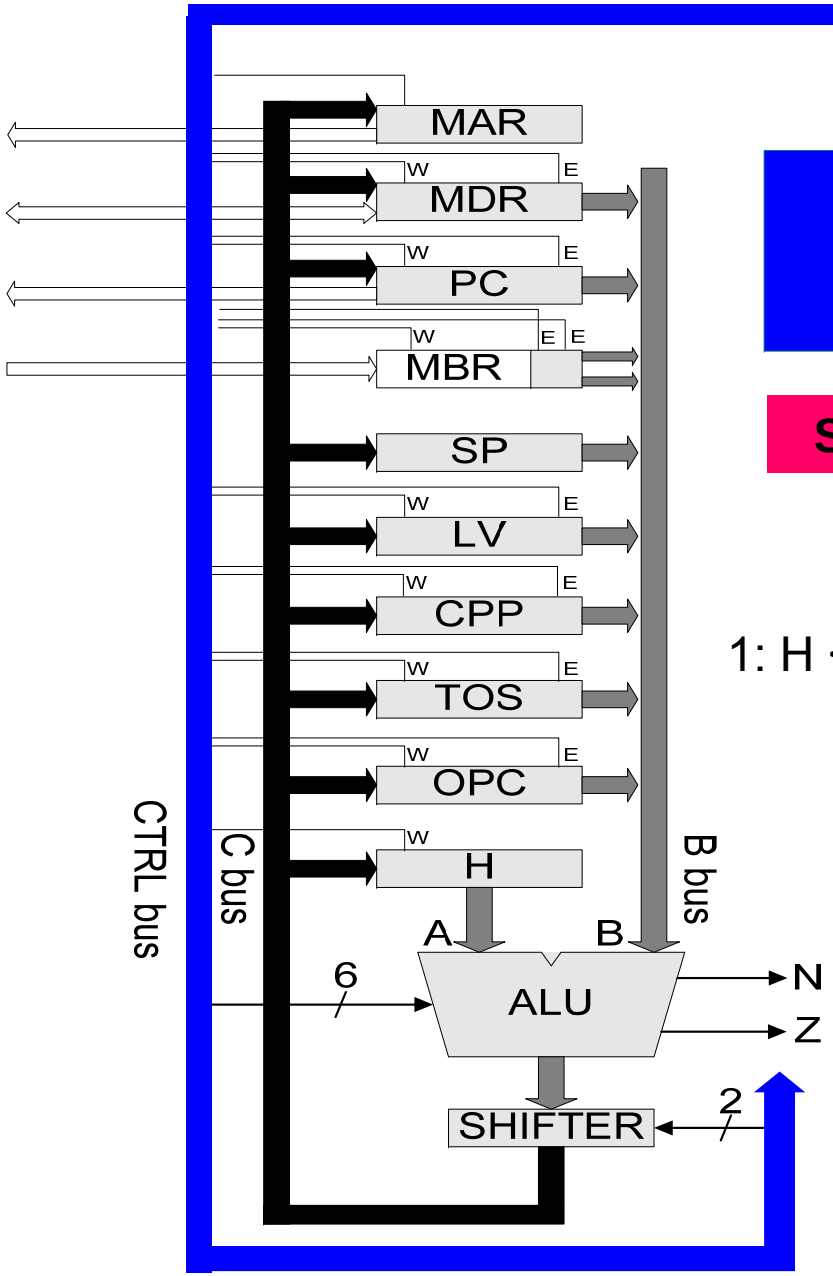




B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Eksempel: 2



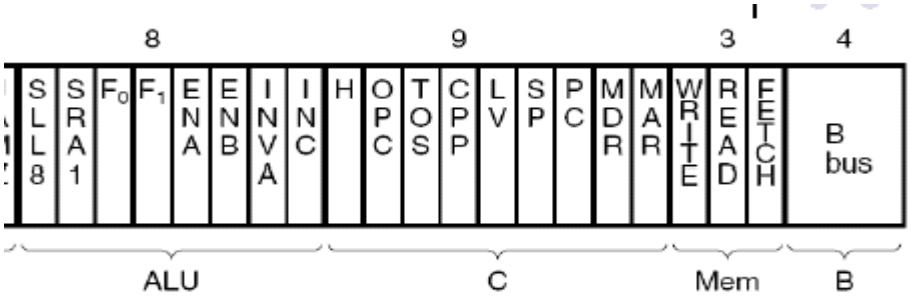
Styreeinheit
Instruction
Decode
Ctrl-logic

SP = TOS + OPC

F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1

1: H <- TOS:

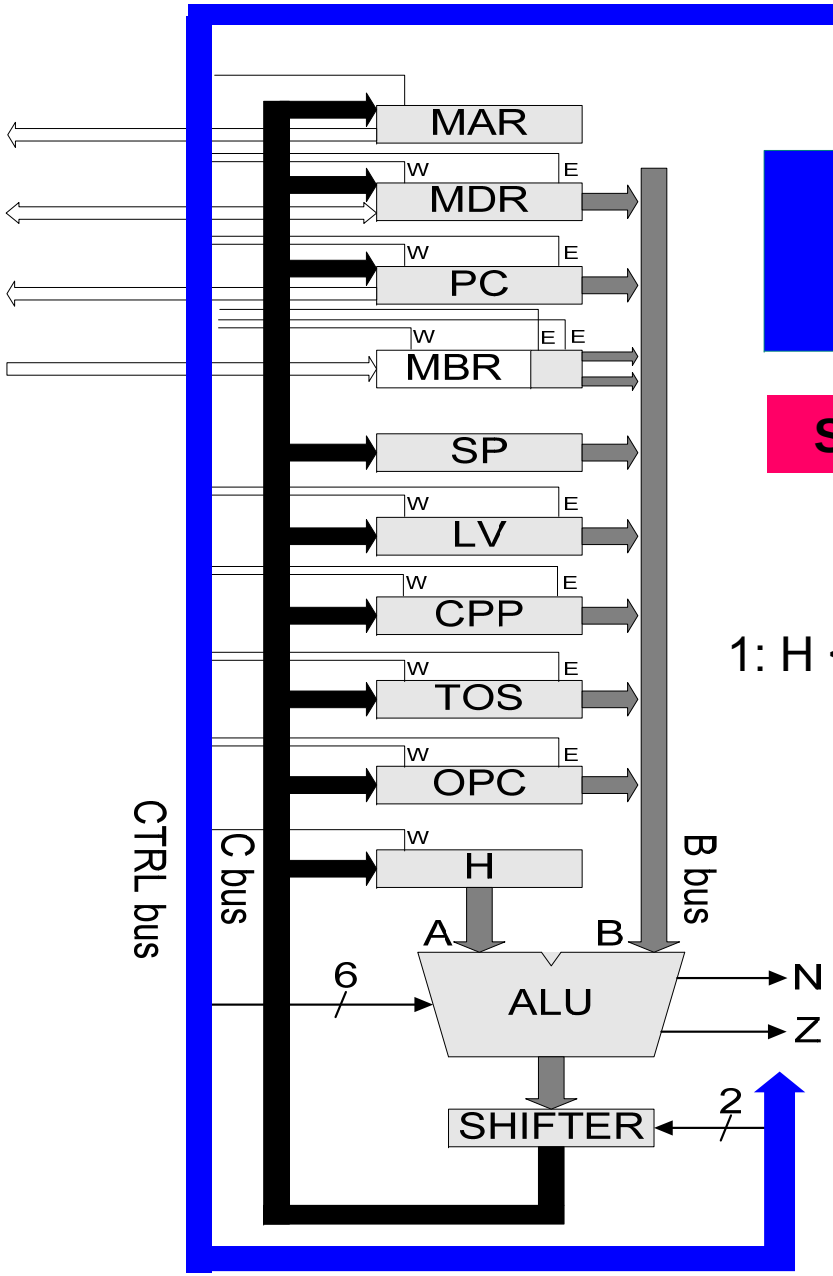
shift	ALU	les frå C buss	mem	B buss
00	010100	XXXXXXXXX	XXX	XXXX



B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Eksempel: 2



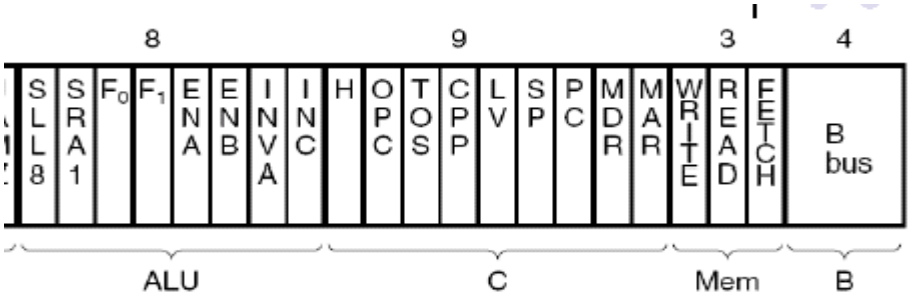
Styreeinheit
Instruction
Decode
Ctrl-logic

SP = TOS + OPC

F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1

1: H <- TOS:

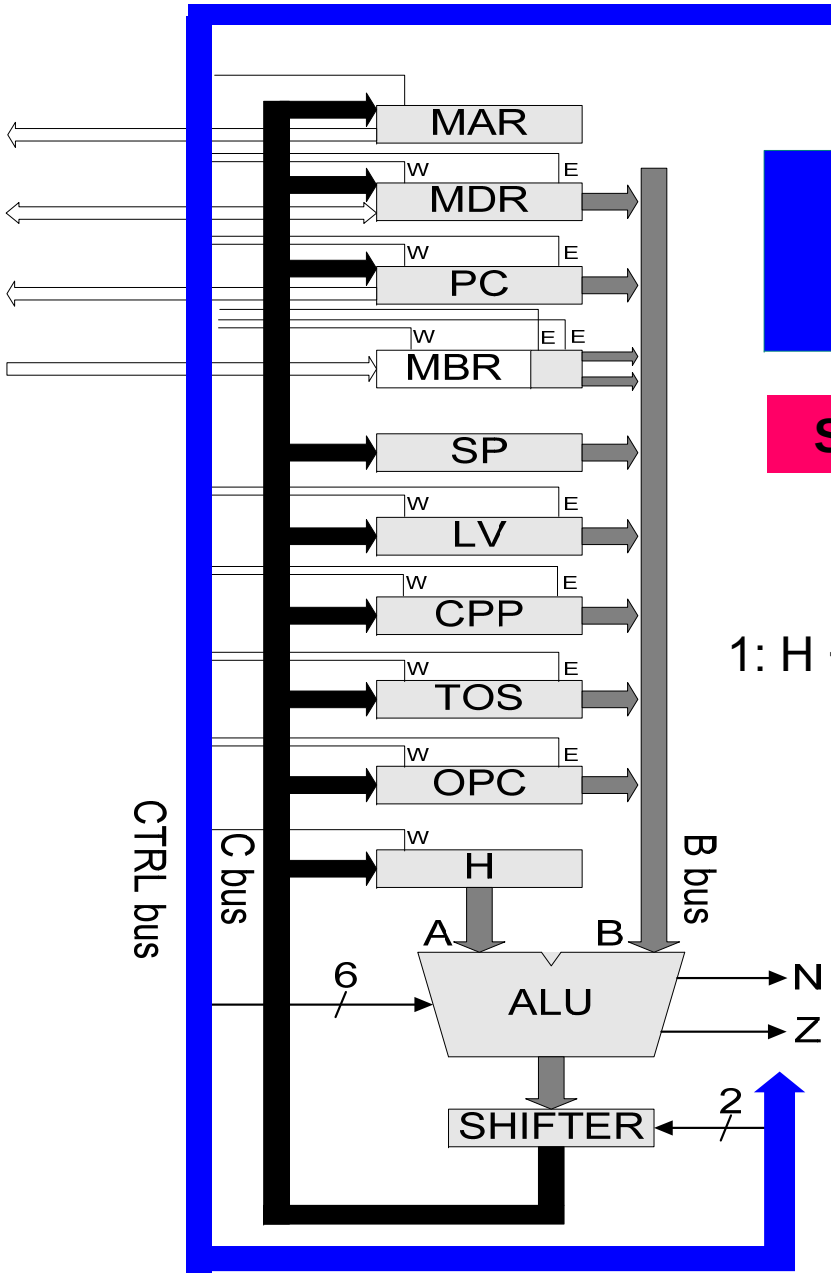
shift	ALU	les frå C buss	mem	B buss
00	010100	1000000000	XXX	XXXX



B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Eksempel: 2



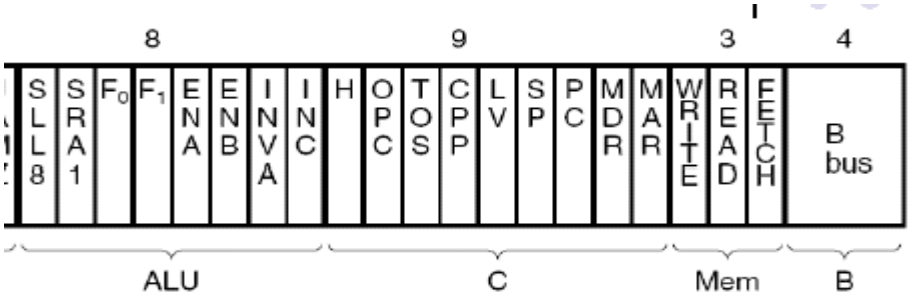
Styreeinheit
Instruction
Decode
Ctrl-logic

SP = TOS + OPC

F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1

1: H <- TOS:

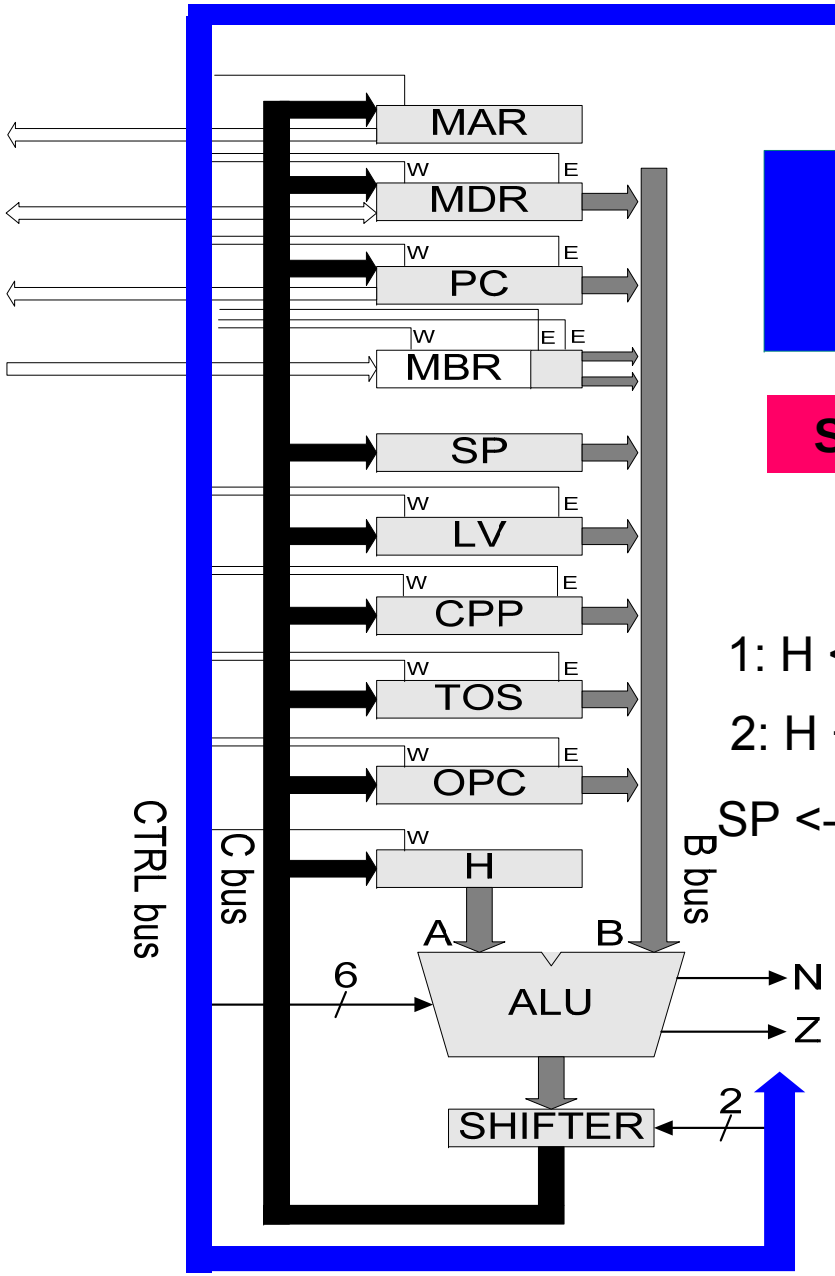
shift	ALU	les frå C buss	mem	B buss
00	010100	1000000000	000	0111



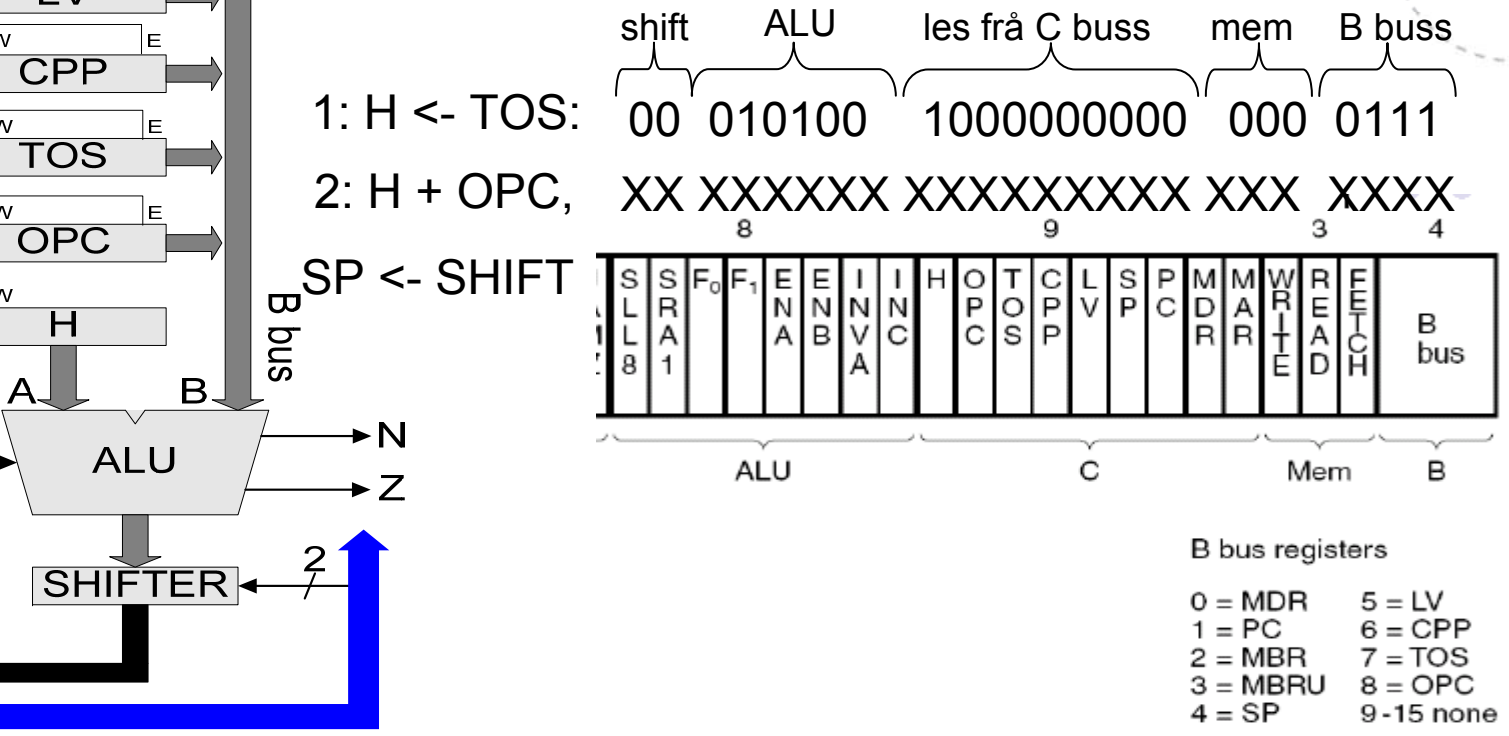
B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Eksempel: 2



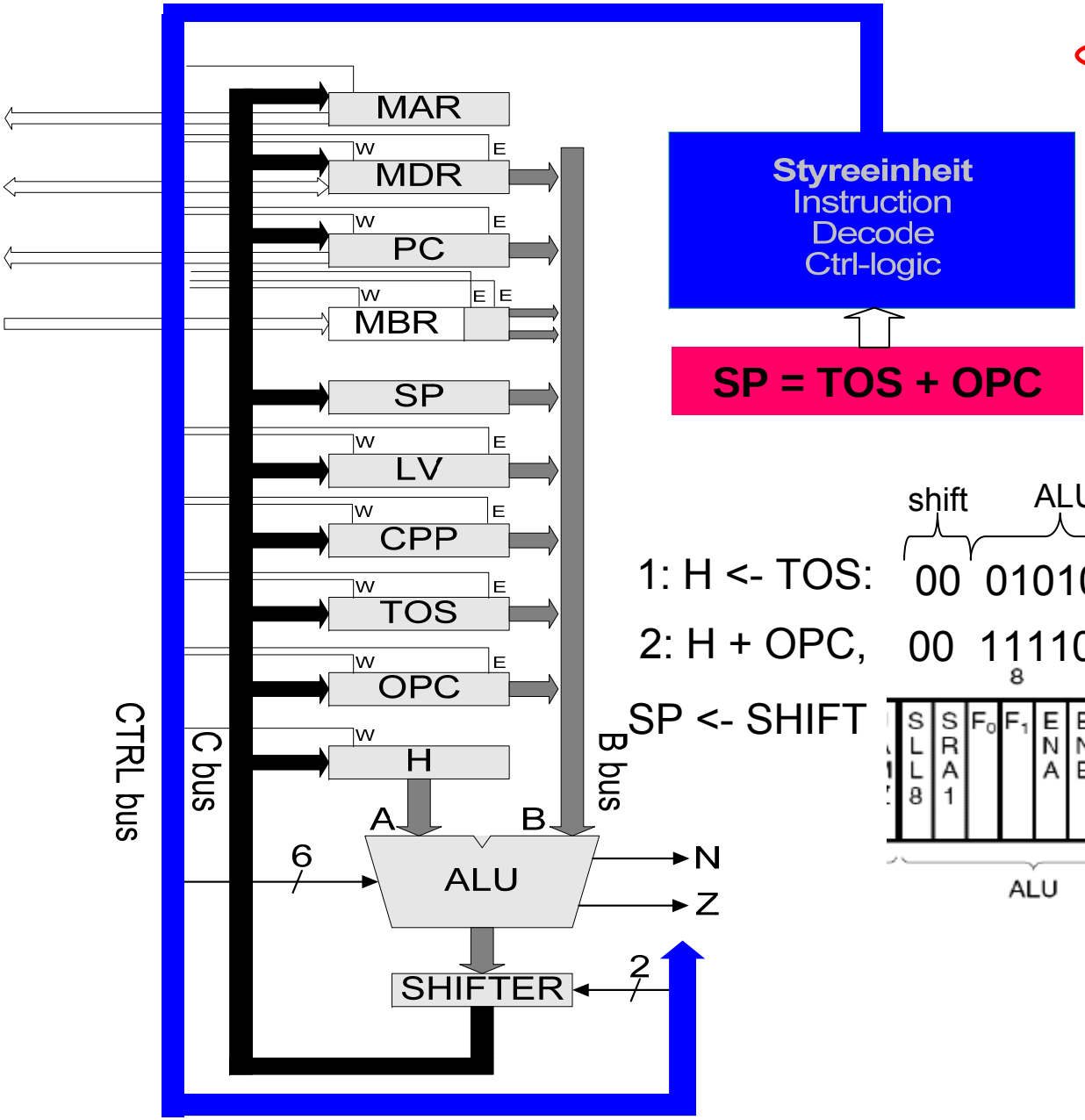
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1



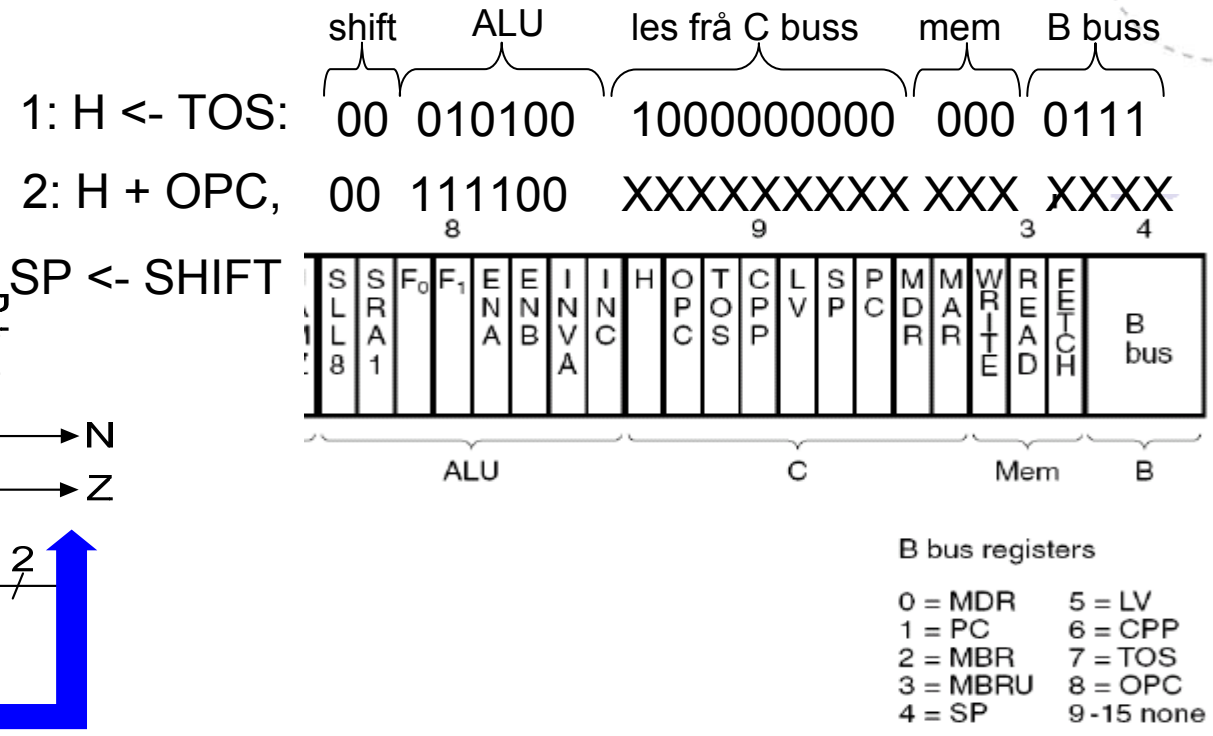
B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Eksempel: 2



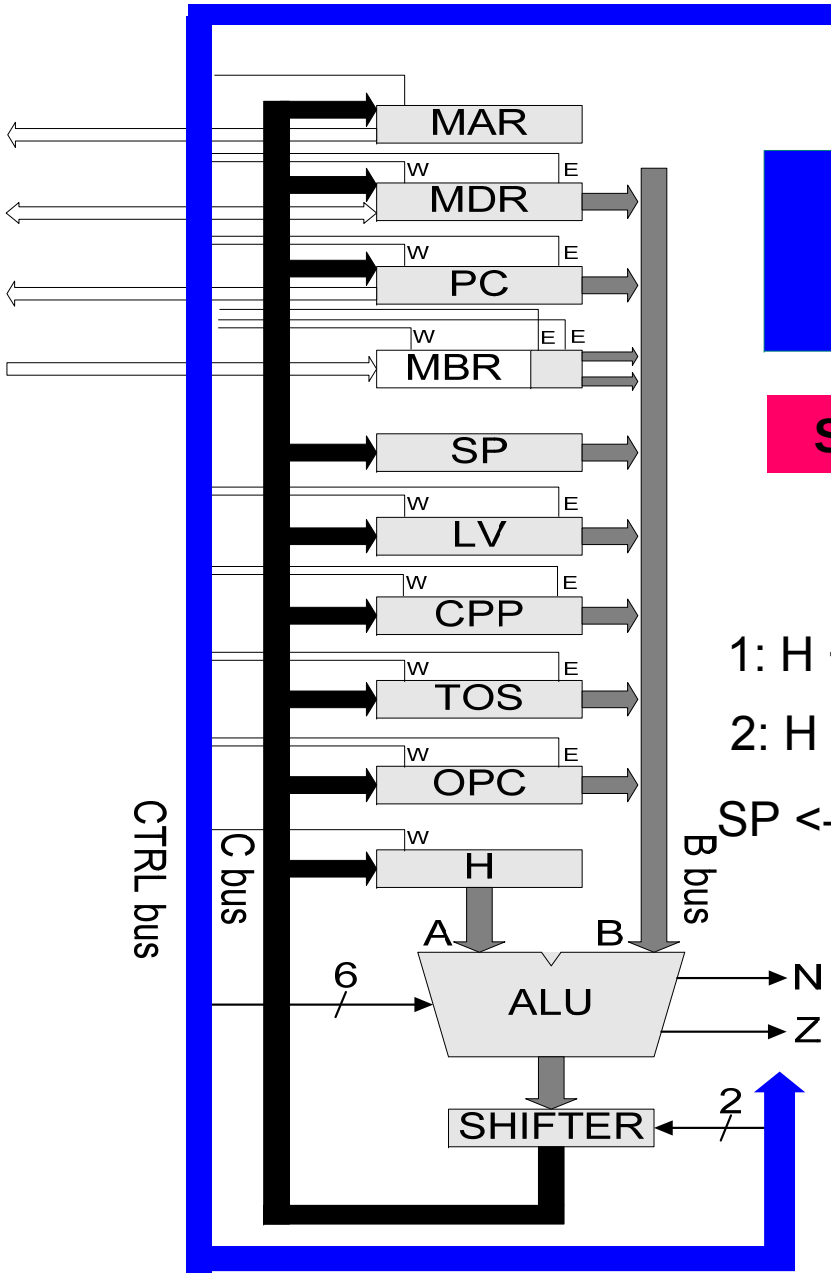
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1



B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Eksempel: 2

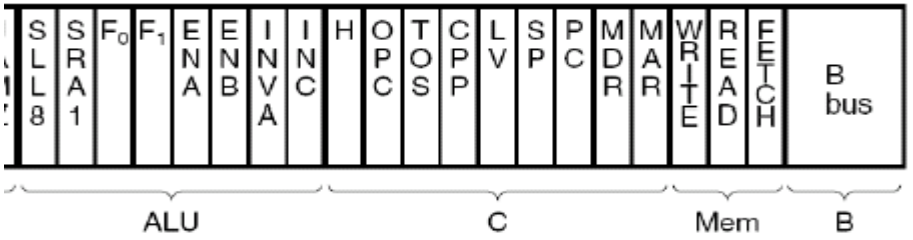


Styreeinheit
Instruction
Decode
Ctrl-logic

SP = TOS + OPC

F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1

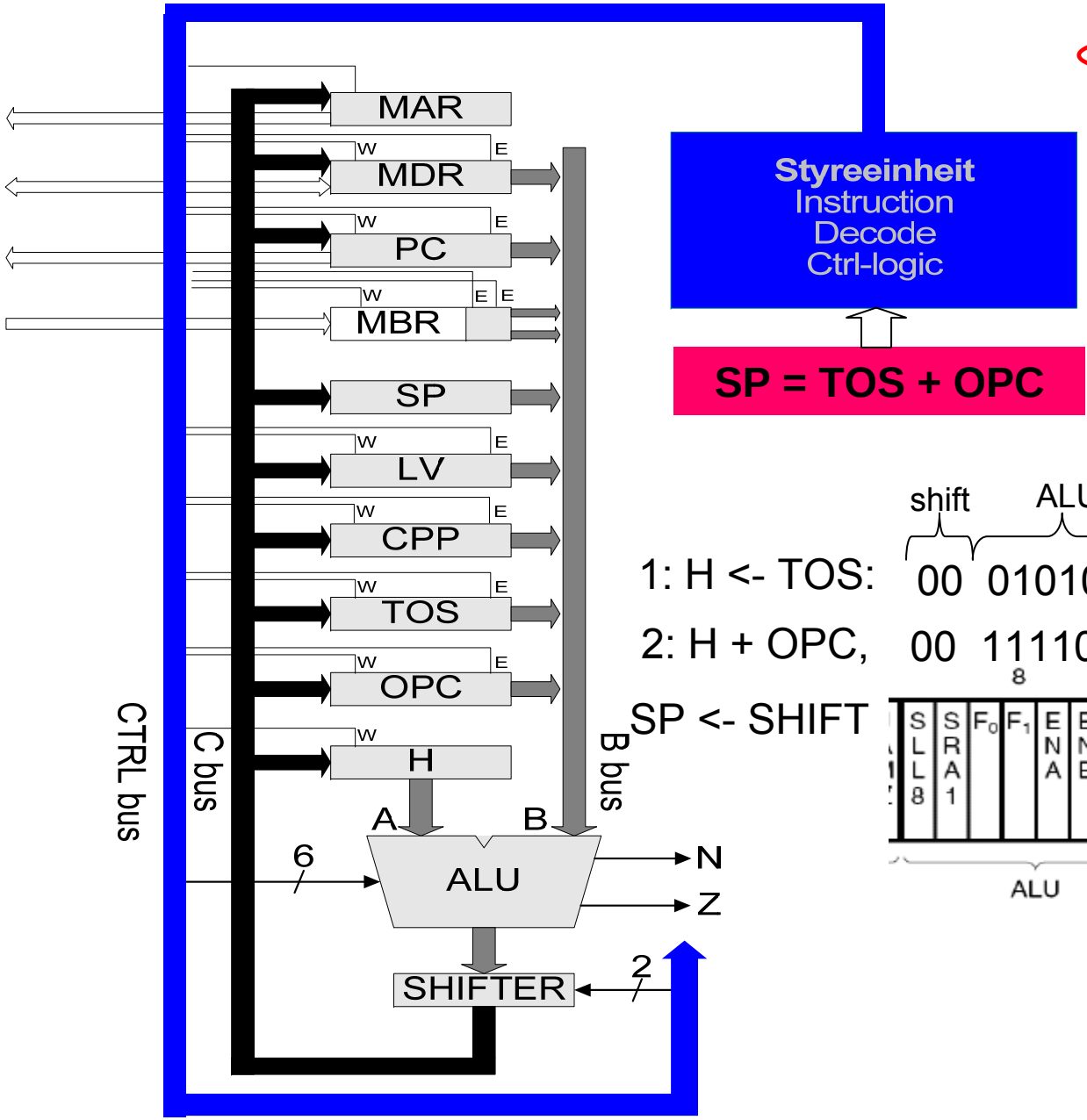
	shift	ALU	les frå C buss	mem	B buss
1: H <- TOS:	00	010100	1000000000	000	0111
2: H + OPC, SP <- SHIFT	00	111100	000001000	XXX	XXXX



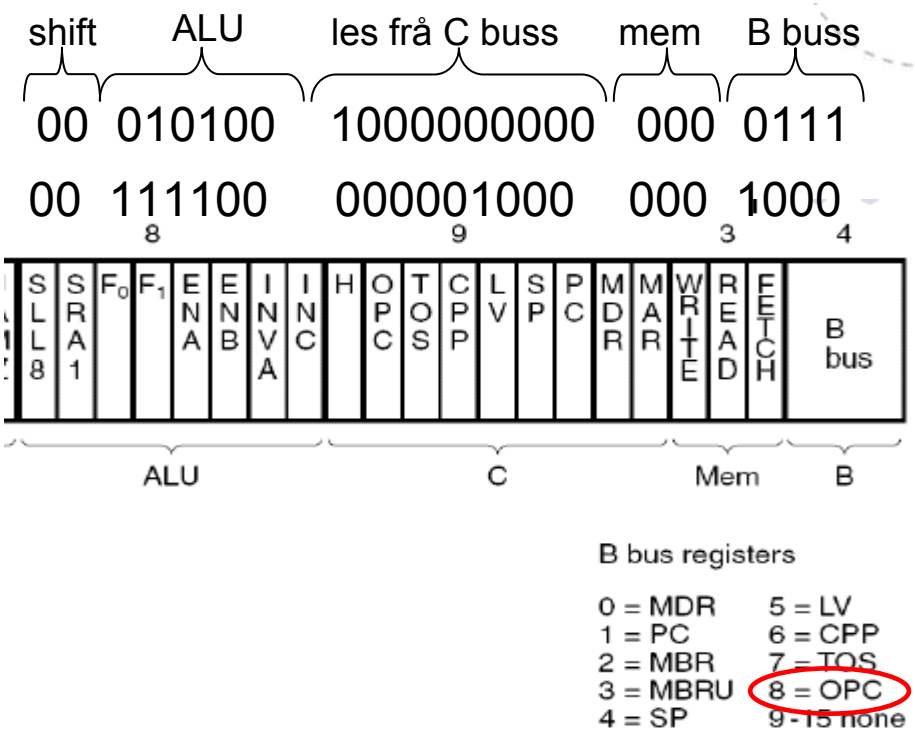
B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Eksempel: 2



F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1

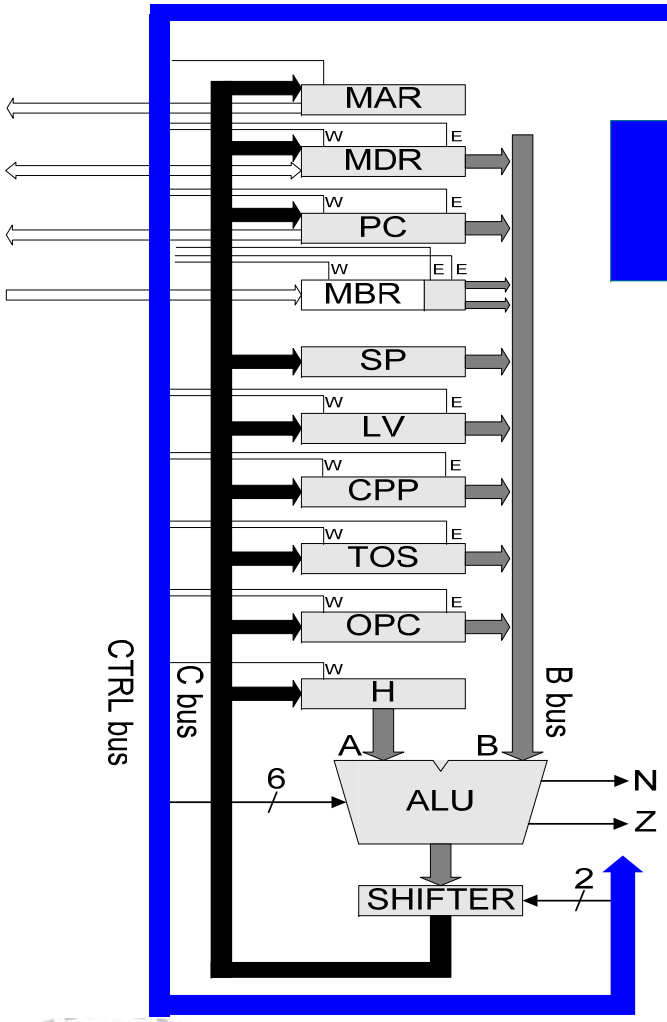


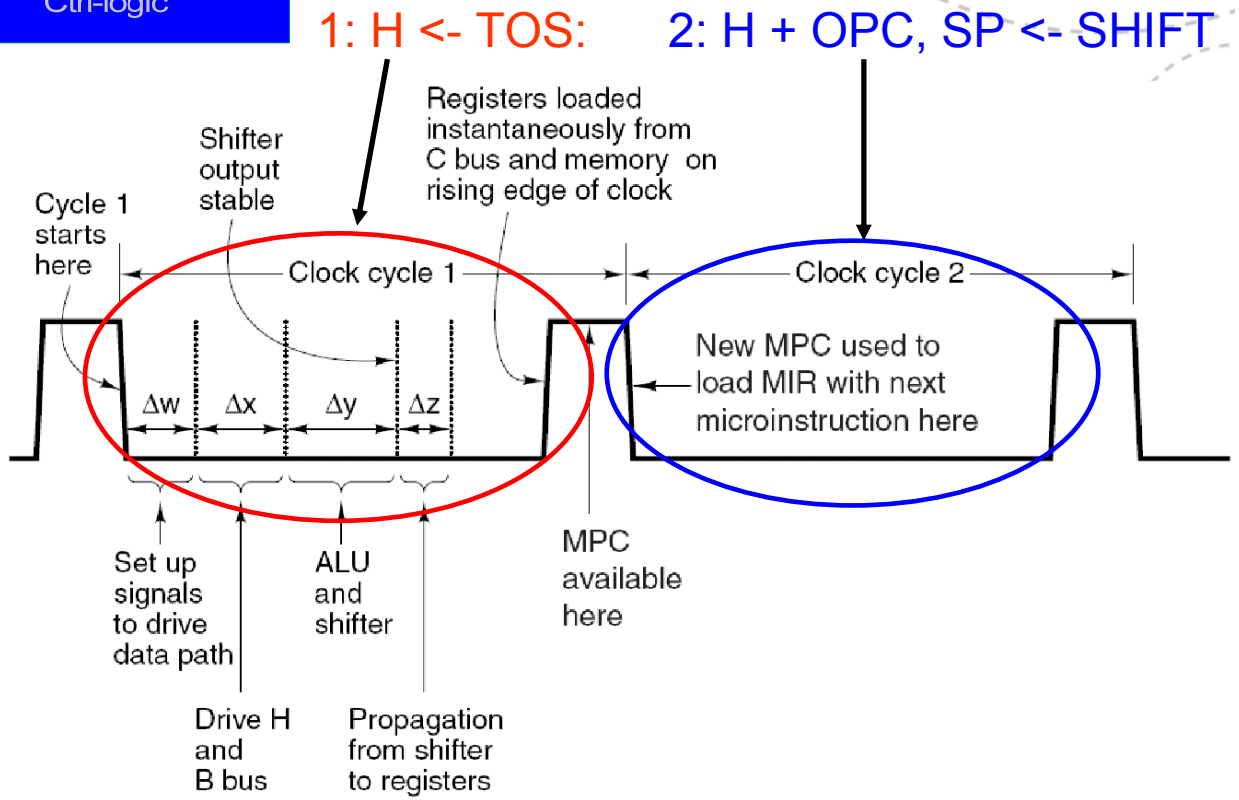
B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Utføre instruksjon

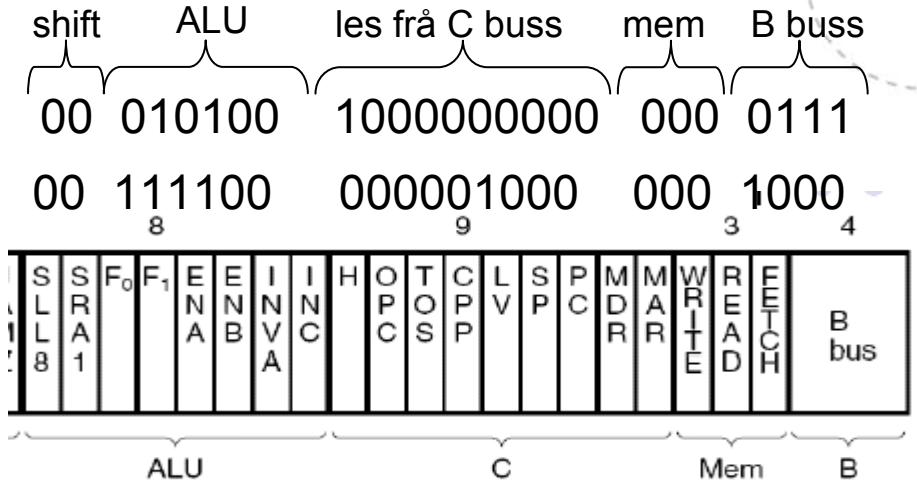
	shift	ALU	les frå C buss	mem	B buss
1:	00	010100	1000000000	000	0111
2:	00	111100	000001000	000	1000





har laga ein microinstruksjon

- **Instruksjon: SP = TOS + OPC**
- ISA er tilgjengeleg for programmerar
  - Viser kva ein instruksjon faktisk gjer på ISA nivå
    - SP = TOS + OPC (eksempel: ADD RegX, RegY, RegZ)
    - Resultat = SP
    - Data = TOS
    - Data = OPC
    - Utførest på to klokkeperiodar
    - Flagg ????



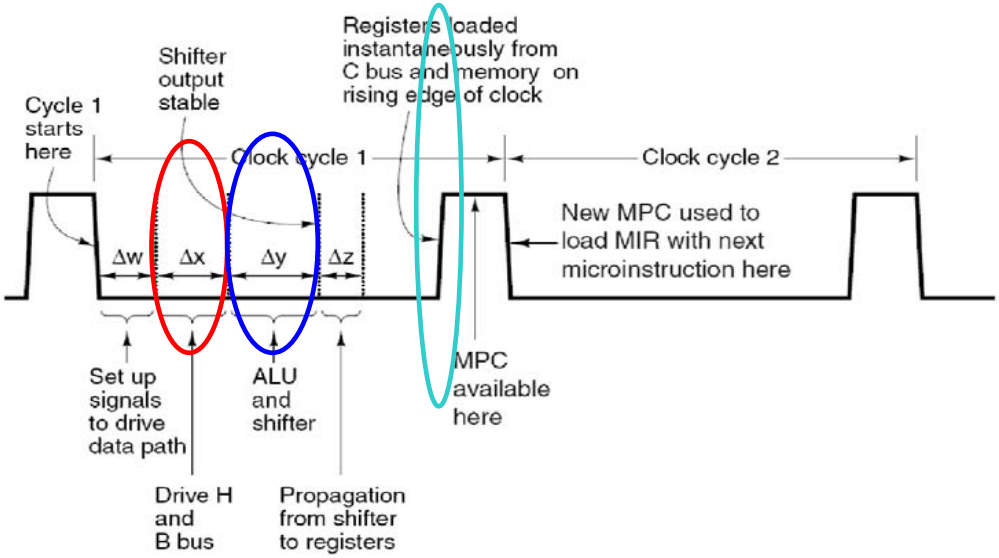
B bus registers

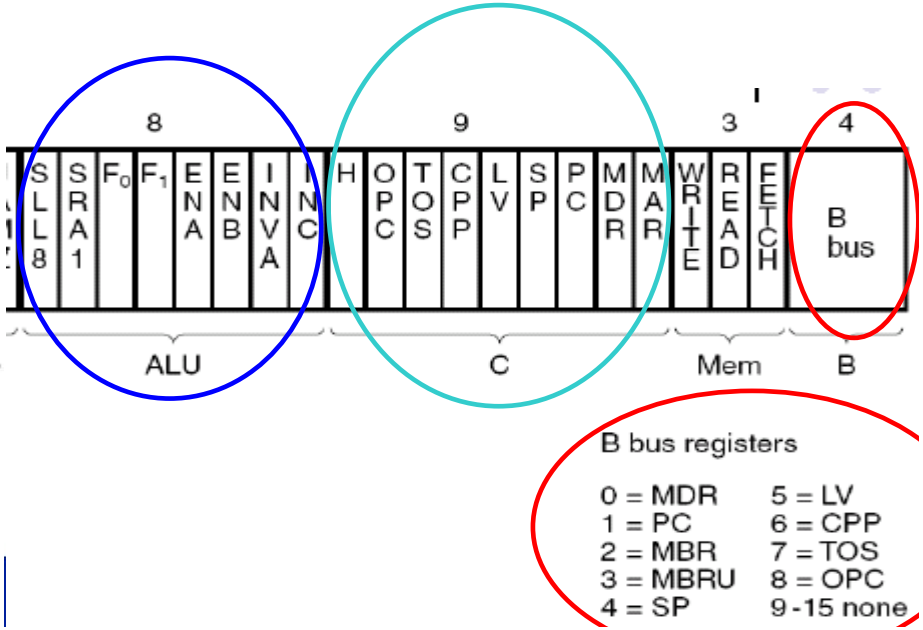
0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

- **På ISA nivå:**
- ADD RegX, RegY, RegZ
  - Reg kan vere
    - Reg X: Alle register
    - RegY: Alle register utanom H
    - RegZ: alle register utanom H

Skrive microinstruksjonar

- Kan instruksjonen utførast på 1 eller fleire klokkeperiodar?
  - Viss ja: fyll ut microinstruksjon
  - Viss nei: finn antal klokkeperiodar utfrå datapath
    - Fyll ut microinstruksjon for kvar klokkeperiode
- Fylle ut microinstruksjon
  - Bruk tidsdiagramme
  - Δw : Setje opp styresignal frå styreeinheit til utførandeeinheitar
  - Δx: Kva skal ut på B bus (H automatisk tilgjengeleg for ALU) FYLL INN B
  - Δy: kva skal ALU og SHIFTER gjere FYLL INN ALU (bruk ALU funksjonstabell)
  - Stigande flanke load register frå C bus: Kva register skal resultatet lagrast i FYLL INN C





Neste gong

- Vidare med microinstruksjonar for IJVM
- Vidare med microarkitektur for IJVM