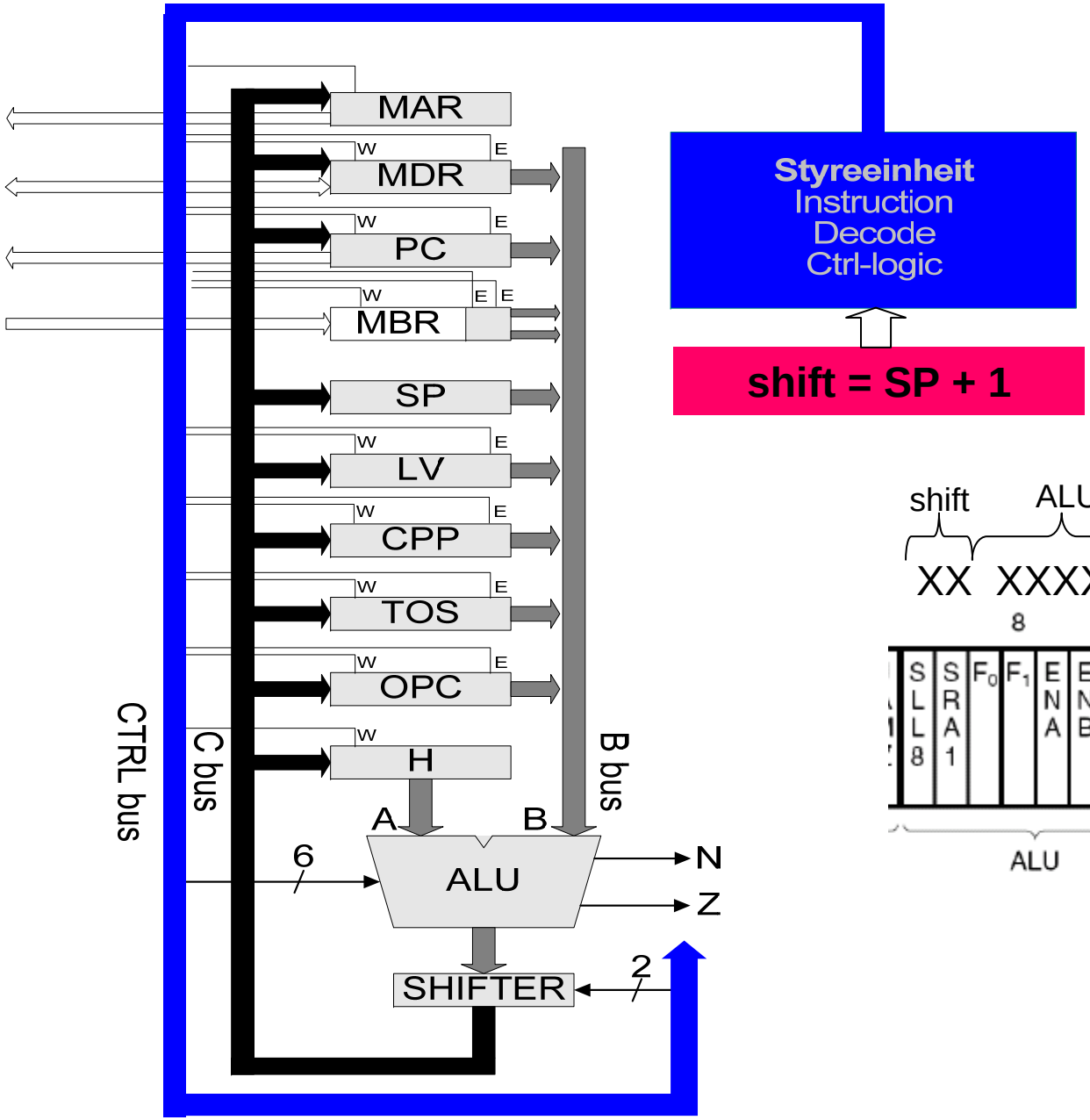
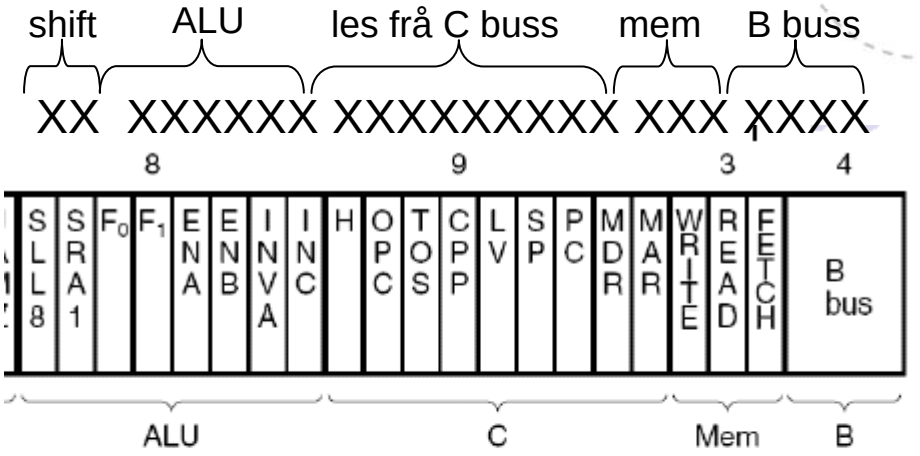
Fortsetelse Microarchitecture level

Lita oppgåve 1:



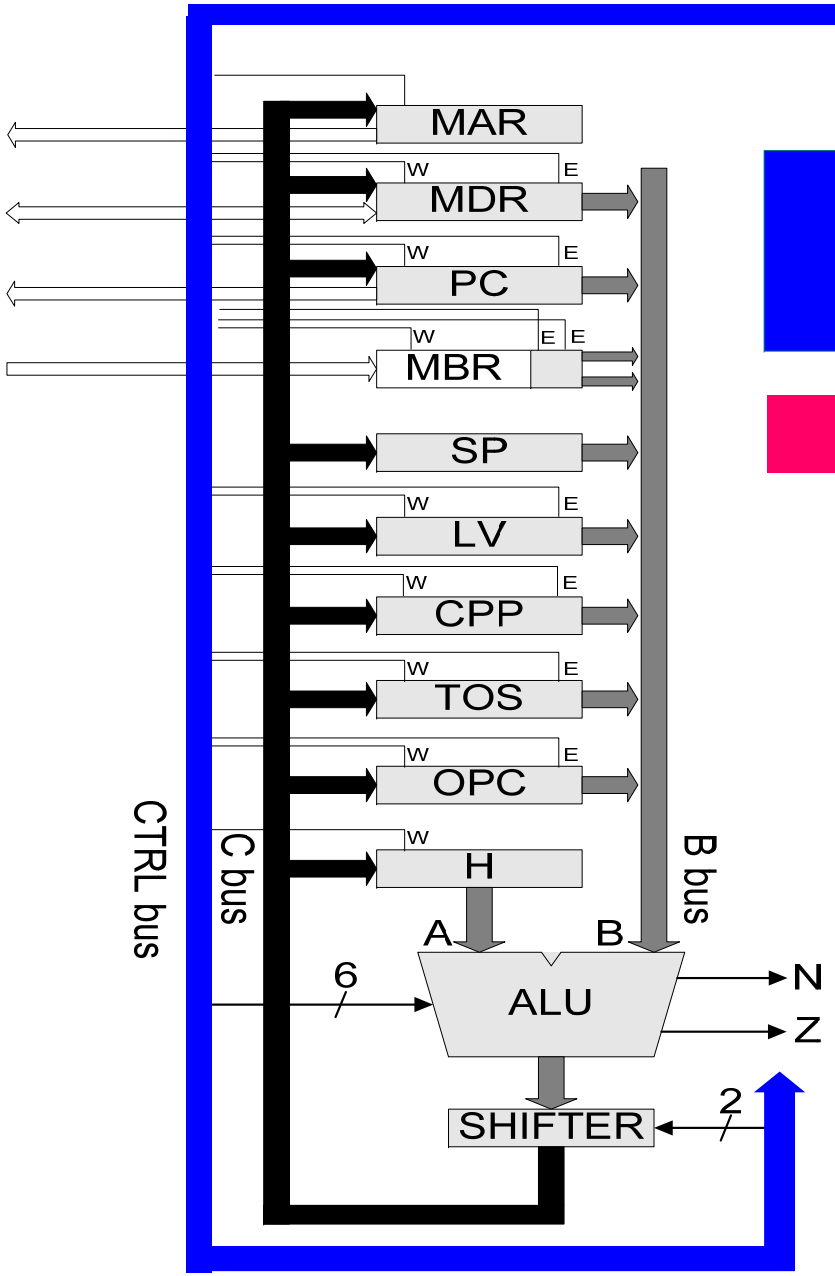
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0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

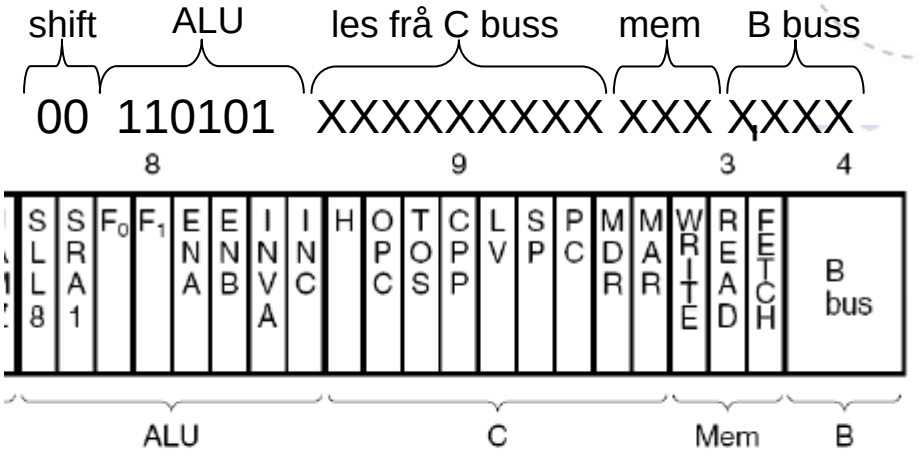
Svar lita oppgåve



Styreeinheit
Instruction
Decode
Ctrl-logic

shift = SP + 1

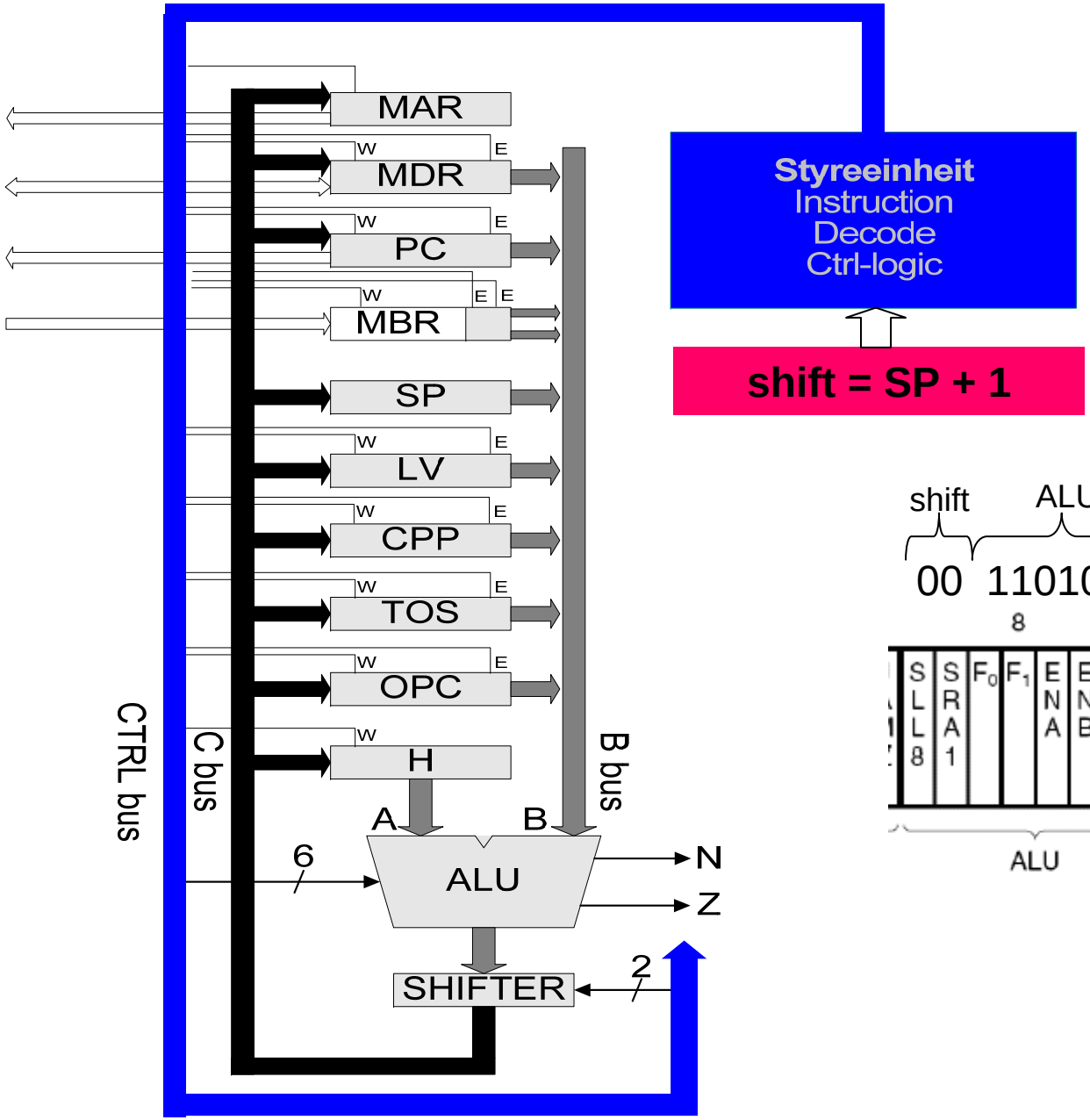
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0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



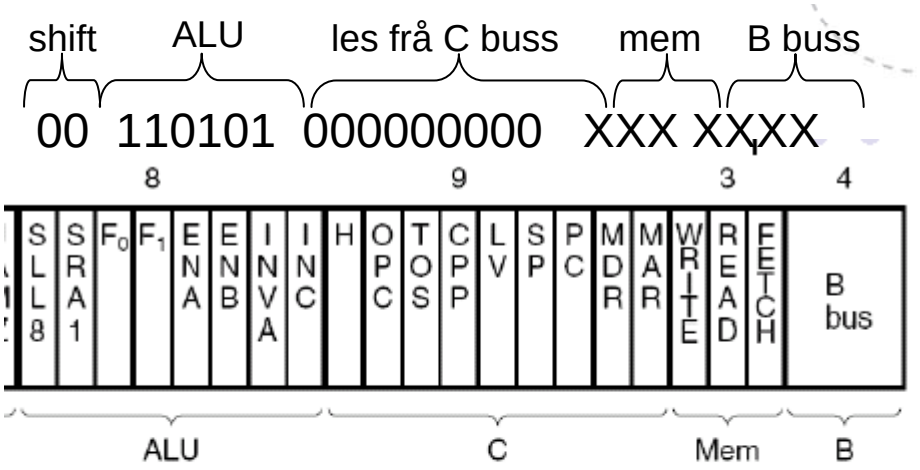
B bus registers

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

Svar lita oppgåve



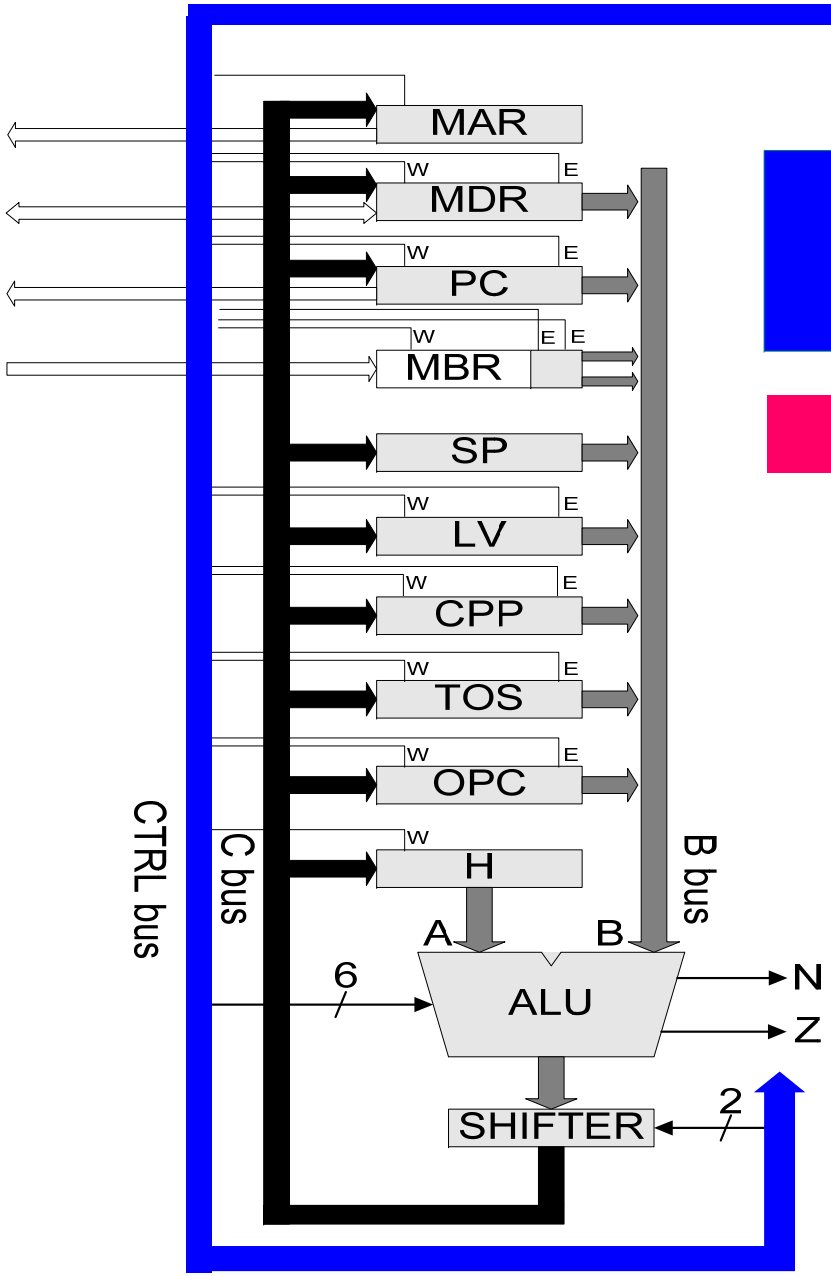
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0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

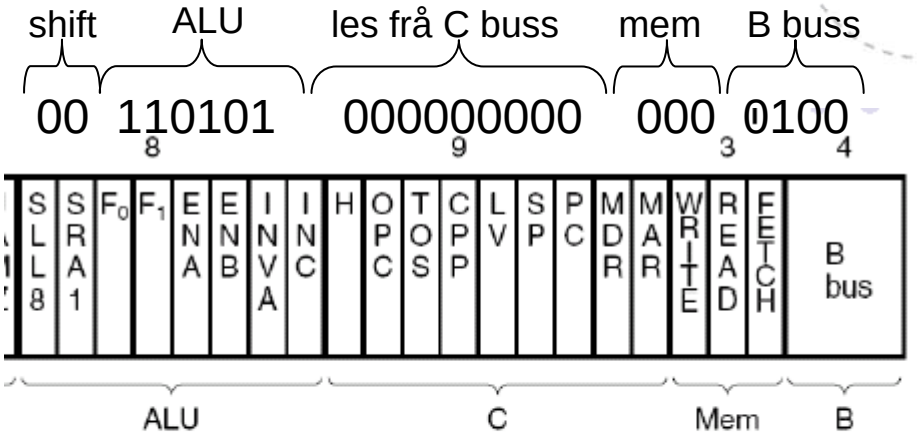
Svar lita oppgåve



Styreeinheit
Instruction
Decode
Ctrl-logic

shift = SP + 1

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

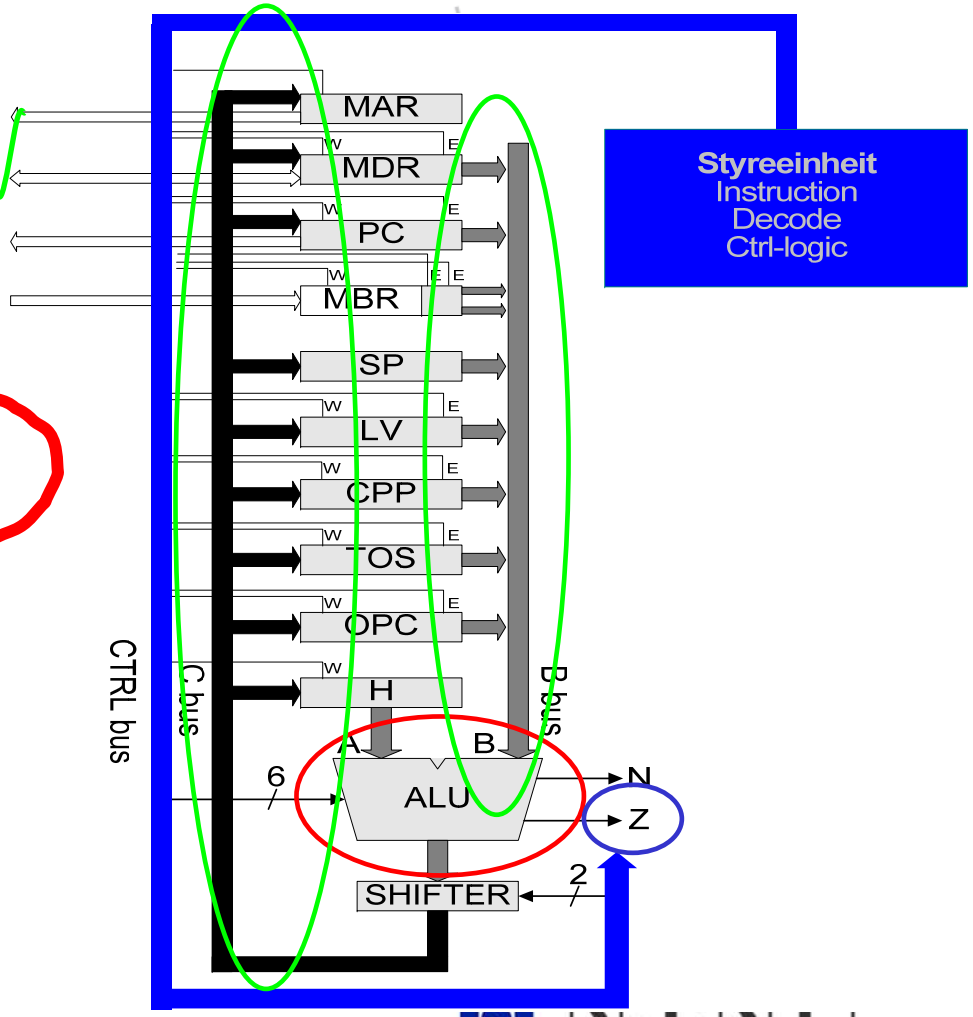
0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Kva kan datamaskiner (frå 1. forelesing)

- Leggje saman to tal √
- Kontrolere om eit tal er 0 √
- Flytte data frå ein plass til ein anna √

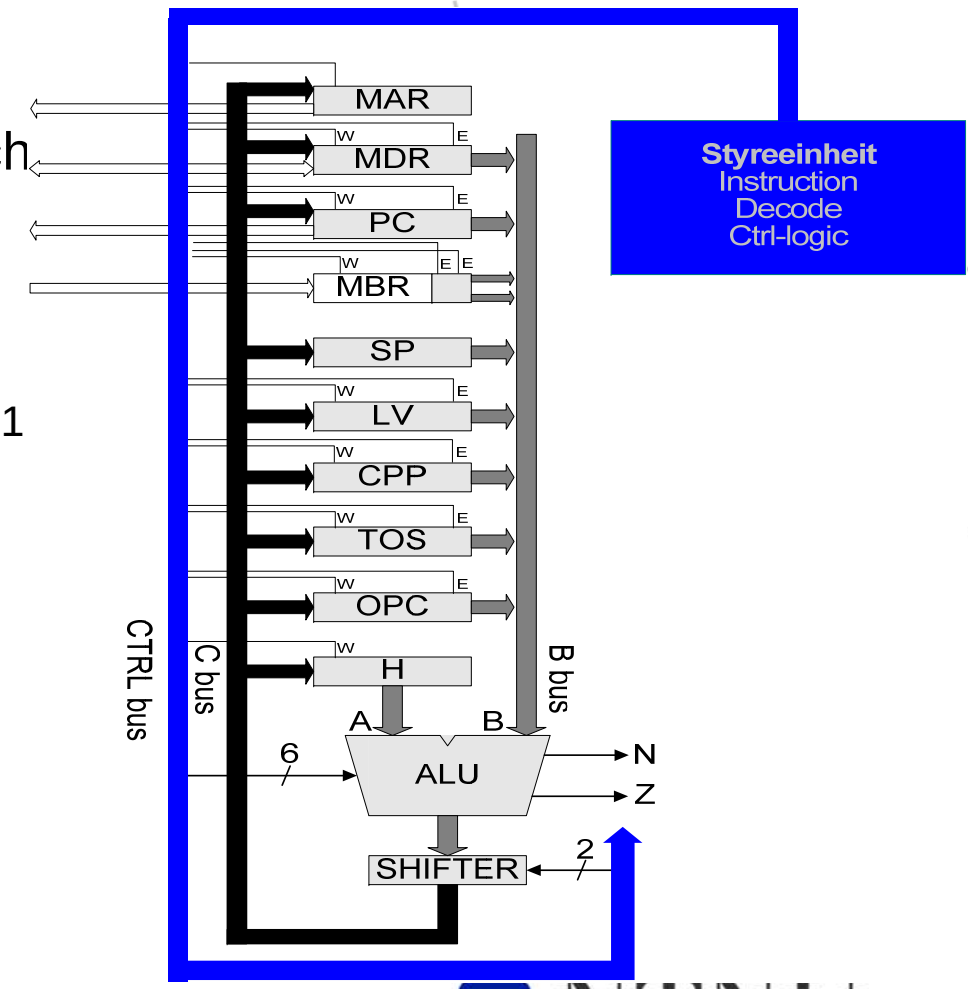
- Gjere desse operasjonane FORT

- Beregne alle beregnbare funksjonar
  - Utføre sekvensar av oprasjonar

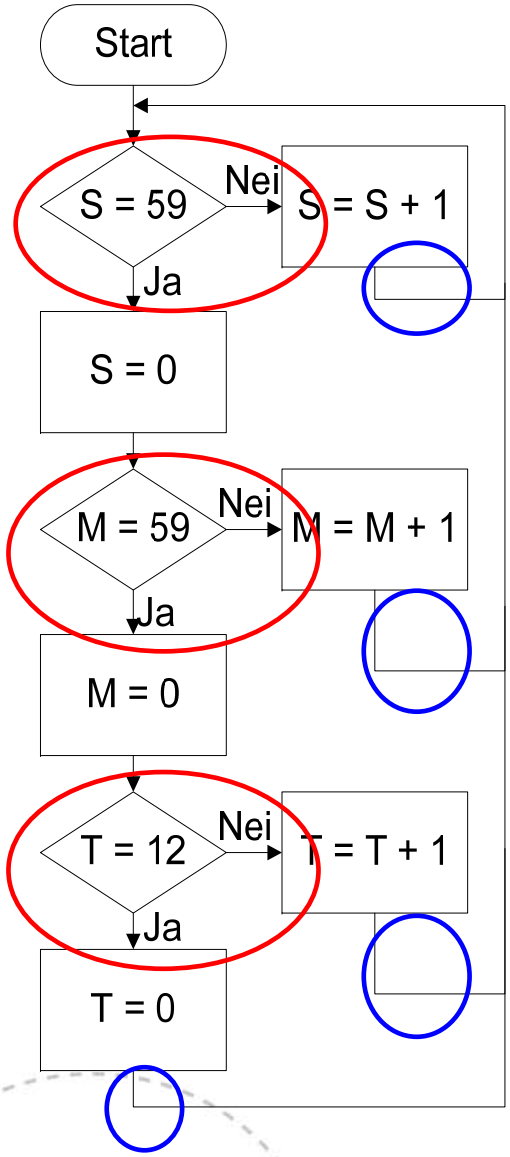


Utføre sekvensar av oprasjonar

- Løkker (loops)
  - Repeter ei rekke instruksjonar
- Ubetinga hopp (Unconditional Branch eller jump)
  - Hoppar alltid (goto adr xxxx)
- Betinga hopp (Conditional Branch)
  - Hoppar viss (Z-flagg = 1 eller N-flagge = 1 goto adr xxxx)



Løkke eksempel klokke



Betinga hopp

Ubetinga hopp

Start: *S = 59? (nei) JMP Sekund*
S = 0
M = 59? (nei) JMP Minutt
M = 0
T = 12 (nei) JMP Time
T = 0
JMP Start

Sekund: *S = S + 1*
JMP Start

Minutt: *M = M + 1*
JMP Start

Time: *T = T + 1*
JMP Start

Løkke eksempel klokke

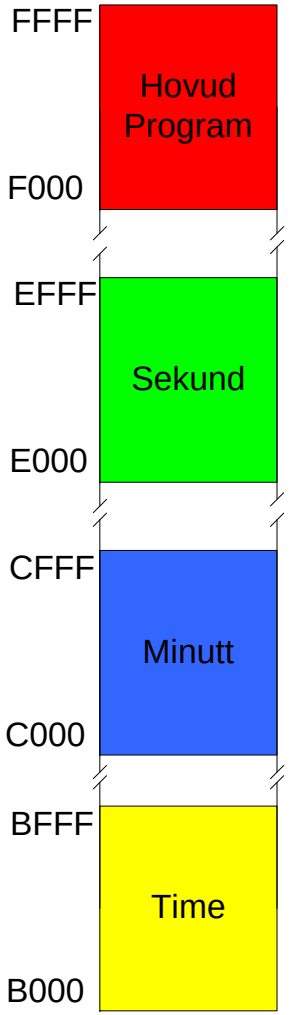
Start:	*S = 59? (nei) JMP Sekund*
	S = 0
	M = 59? (nei) JMP Minutt
	M = 0
	T = 12 (nei) JMP Time
	T = 0
	JMP Start

Sekund:	*S = S + 1*
	JMP Start

Minutt:	*M = M + 1*
	JMP Start

Time:	*T = T + 1*
	JMP Start

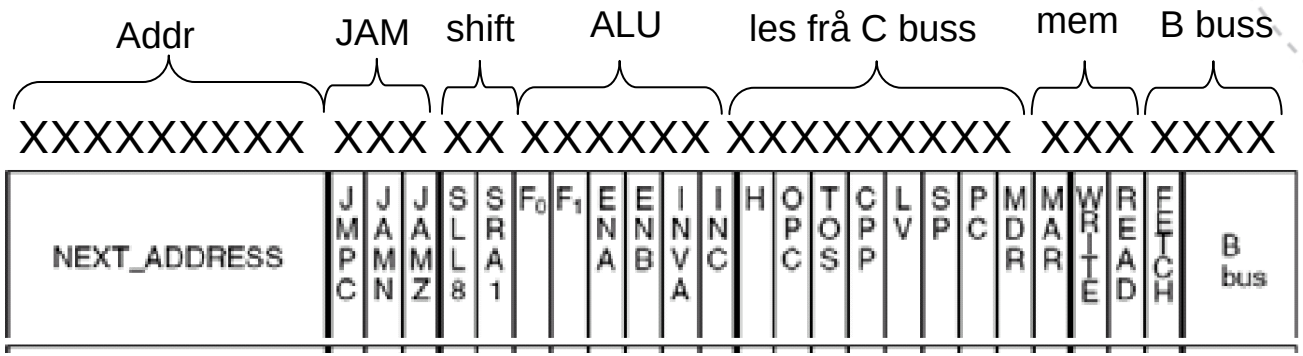
I programminne:



NTNU
Innovation and Creativity

Microinstruksjon Addr feltet

- **Addr gir addressa til neste microinstruksjon**



- Addr peikar på addressa til neste microinstruksjon i instruksjonen
  - Adr 0: Instruksjon 1
    - 1. microinstruksjon ligg på Addr 0, microinstruksjon Addr peikar på Addr 1
    - 2. microinstruksjon ligg på Addr 1, microinstruksjon Addr peikar på 2
    - 3. og siste microinstruksjon ligg på Addr 3, Addr peikar på start ny instruksjon

NTNU
Innovation and Creativity

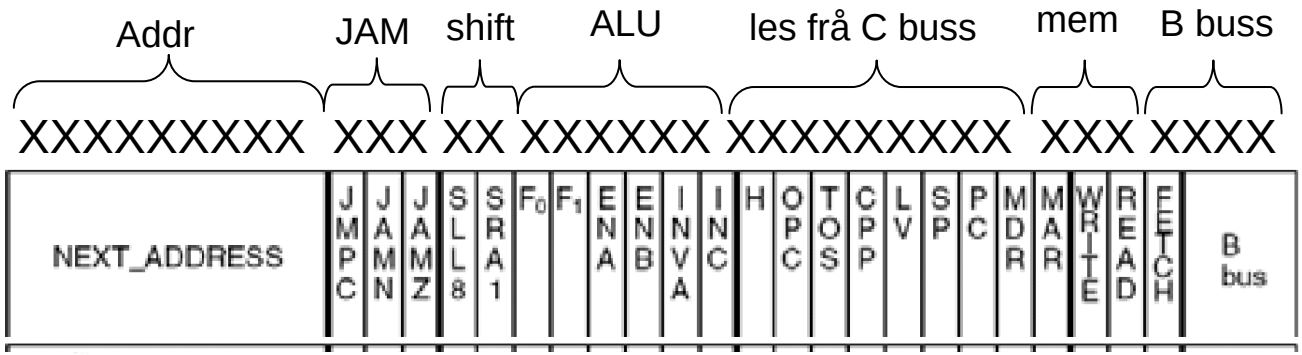
Microinstruksjon Addr feltet

- **Addr gir addressa til neste microinstruksjon**
- **Instruksjon: SP = TOS + OPC (forrige forelesing)**

	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	000	1000

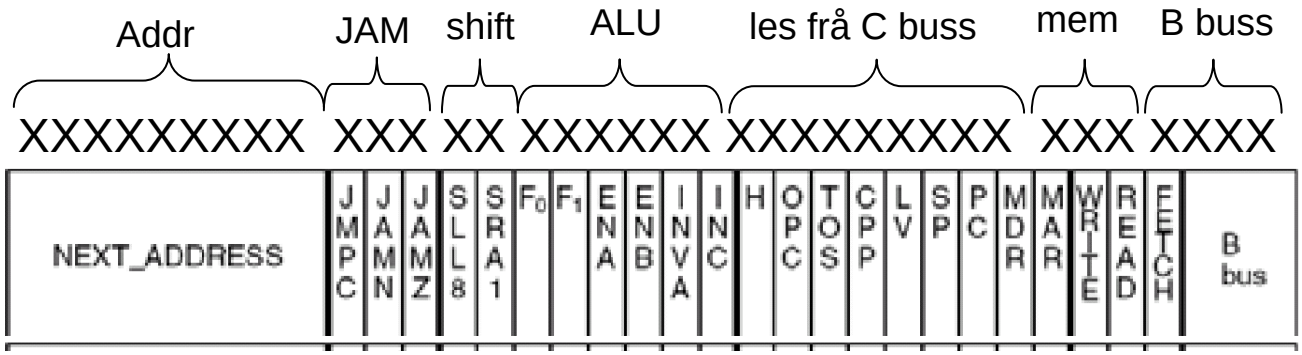
- Addr kan då sjå slik ut (H <- TOS ligg på Addr 0):

	Addr	shift	ALU	les frå C buss	mem	B buss
1: H <- TOS:	000000001 xxx	00	01010	1000000000	000	0111
2: H + OPC, SP <- SHIFT:	000000002 xxx	00	111100	000001000	000	1000

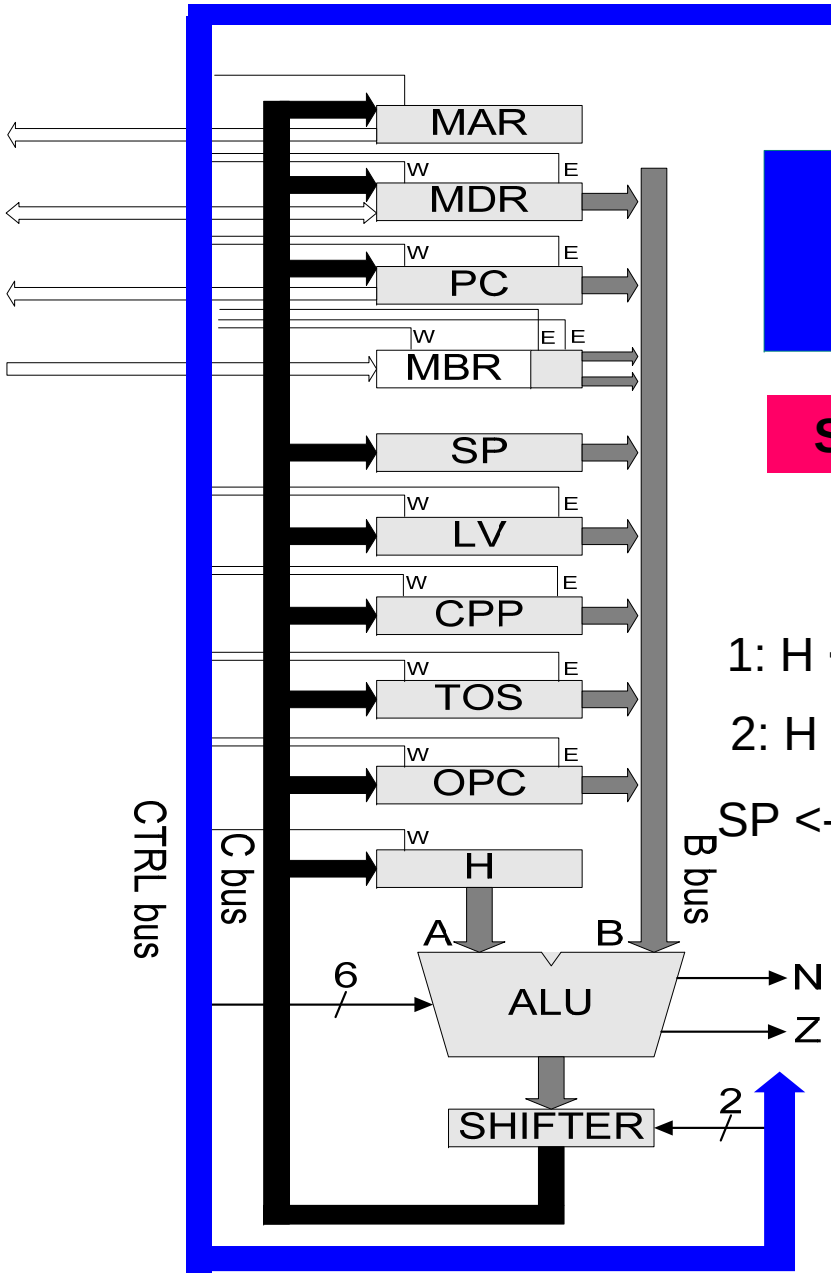


Må ha hopp inn i microinstruksjon

- **Utvidar med JAM**
  - JAMZ: Sjekkar Z-flagget (zero) til ALU
  - JAMN: Sjekkar N-flagget (negativ) til ALU
  - JMPC: Ekstra hoppting kjem seinare
- No mogleg å detektera N og Z for å kunne bestemme hopp
  - Kan sjekke resultat frå beregning utført av ALU
  - Kan bruke resultatet til betingahopp



Fleire microinstr. for instr.



Styreeinheit
Instruction
Decode
Ctrl-logic

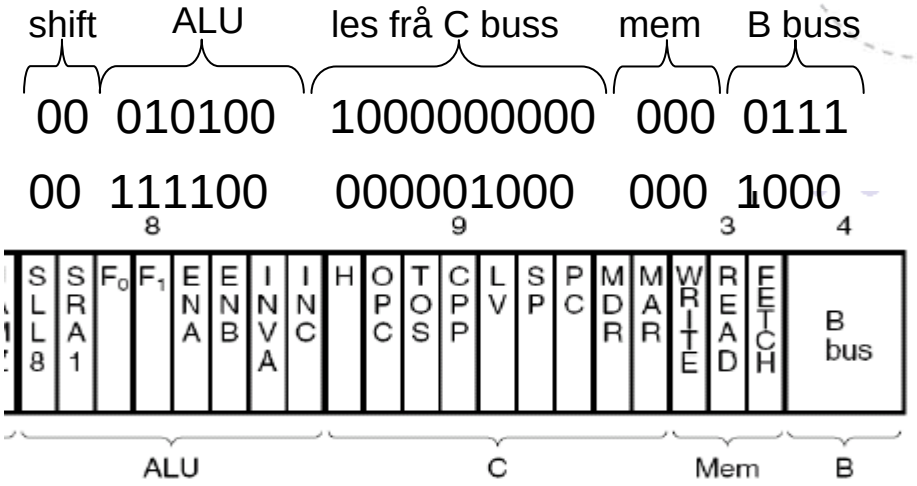
SP = TOS + OPC

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:

2: H + OPC,

SP <- SHIFT



B bus registers

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