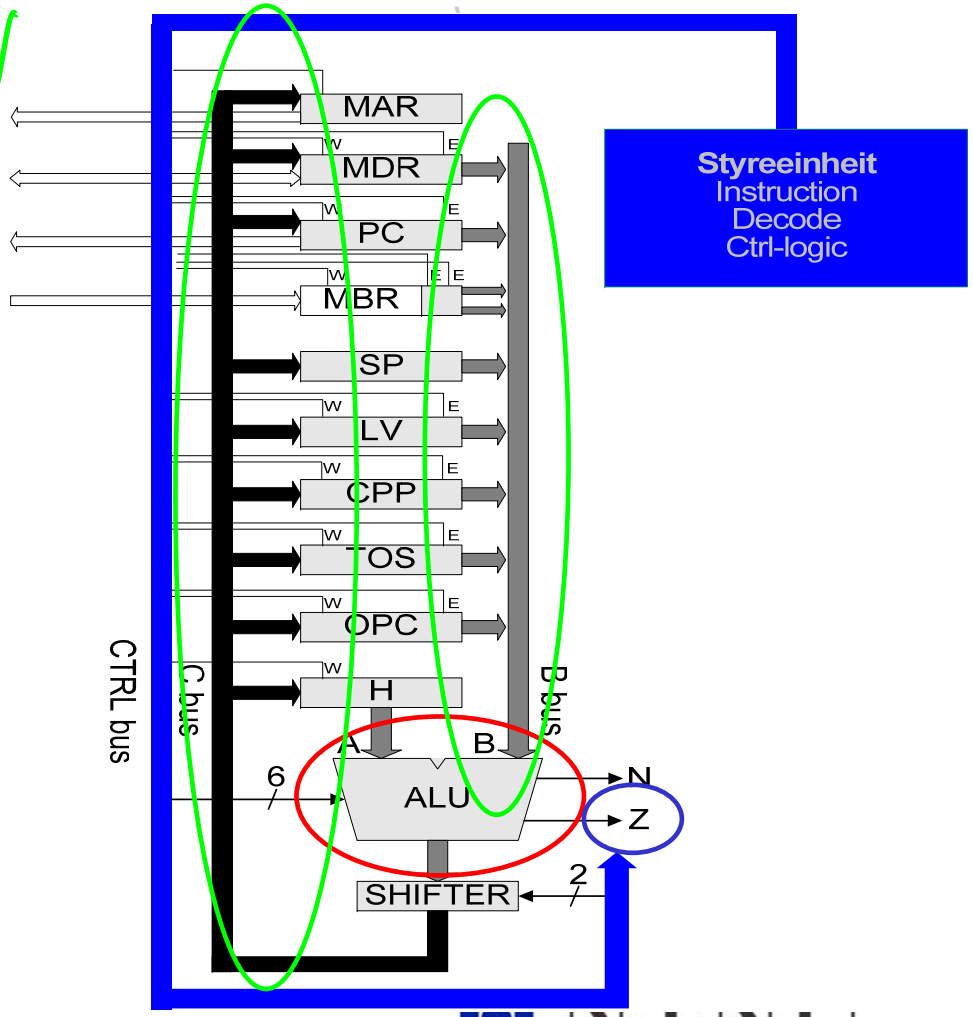
Fortsetelse Microarchitecture level

Kva kan datamaskiner (frå 1. forelesing)

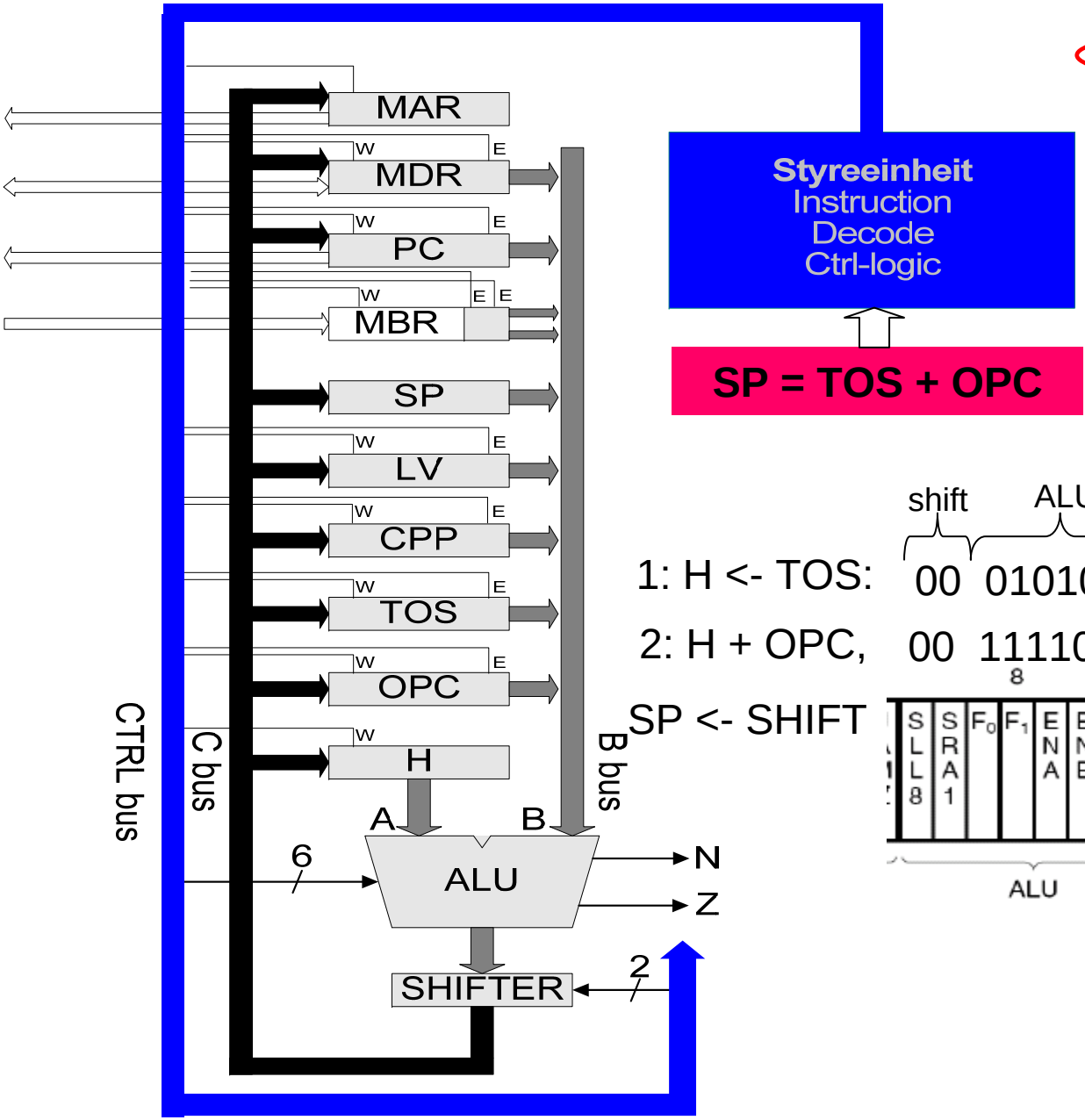
- Aritmetiske Logiske funksjonar √
- Flytte data frå ein plass til ein anna √
- Test er eit tal 0? √

- Gjere desse operasjonane FORT

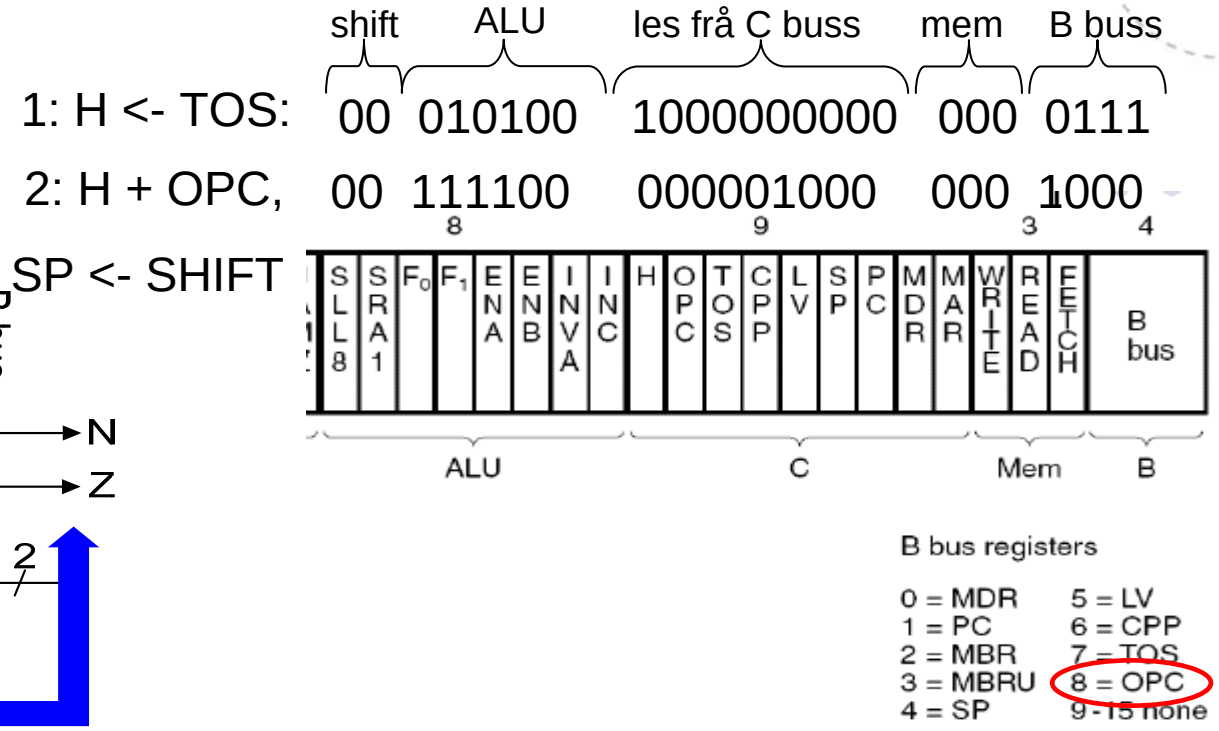
- Berekne alle beregnbare funksjonar
  - Utføre sekvensar av operasjonar



ALU og Flytt/kopier



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



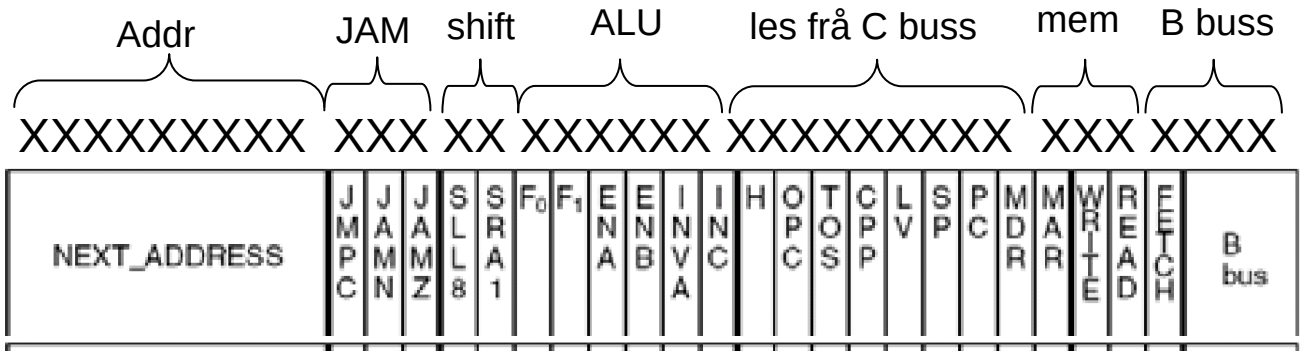
B bus registers

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

Microinstruksjon: ALU og Flytt/kopier

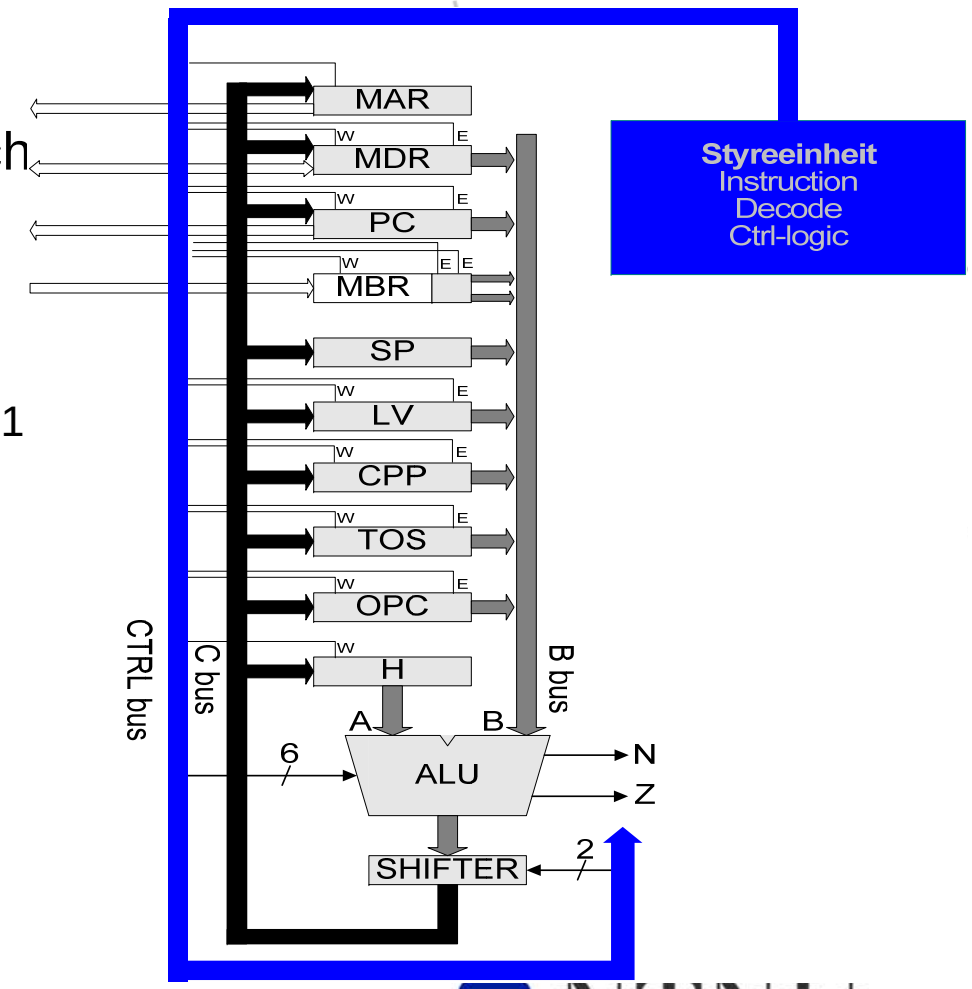
- **Instruksjon: SP = TOS + OPC**

	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



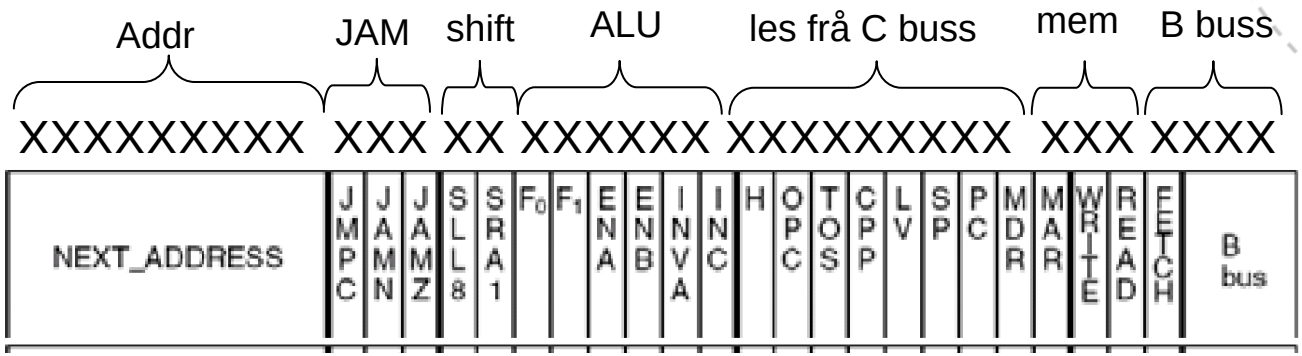
Utføre sekvensar av oprasjonar

- 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)



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

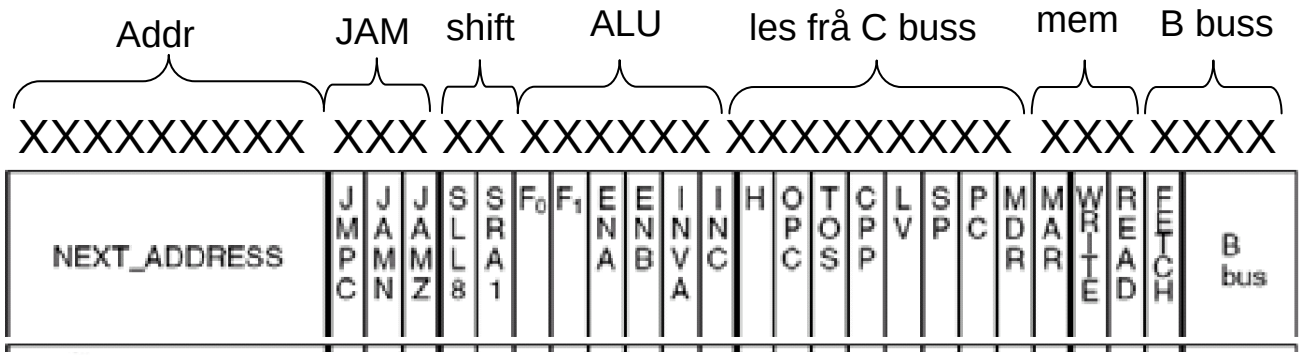
Microinstruksjon Addr feltet

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

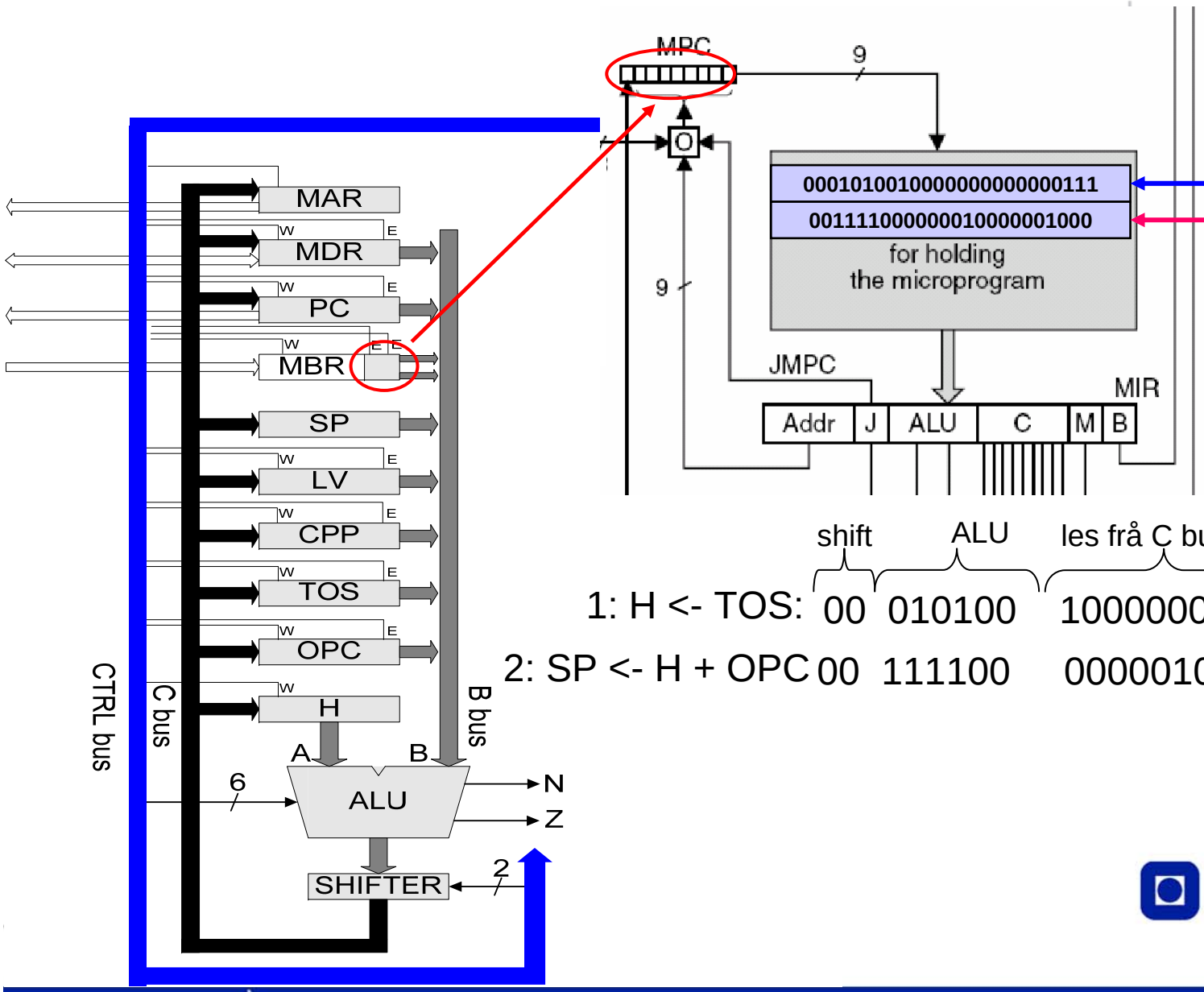
	shift	ALU	les frå C buss	mem	B buss
1: H <- TOS:	00	010100	1000000000	000	0111
2: H + OPC, SP <- SHIFT:	00	111100	000001000	000	1000

- 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



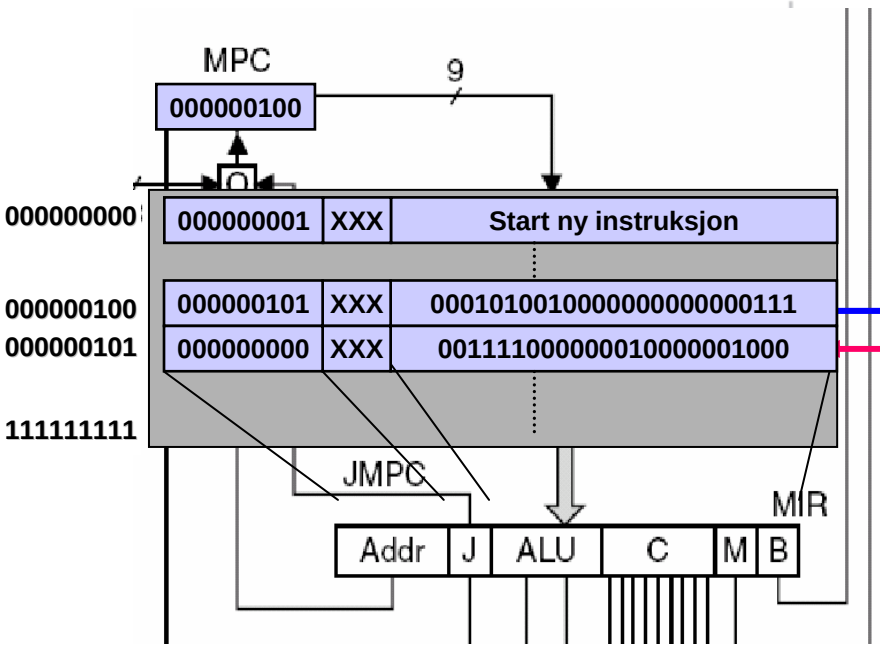
Instr. i MicroProg minne



	shift	ALU	les frå C buss	mem	B buss
1: H <- TOS:	00	010100	1000000000	000	0111
2: SP <- H + OPC	00	111100	000001000	000	1000

NTNU
Innovation and Creativity

Instr. i MicroProg minne



	shift	ALU	les frå C buss	mem	B buss
1: H <- TOS:	00	010100	1000000000	000	0111
2: SP <- H + OPC	00	111100	000001000	000	1000

Løkke eksempel klokke

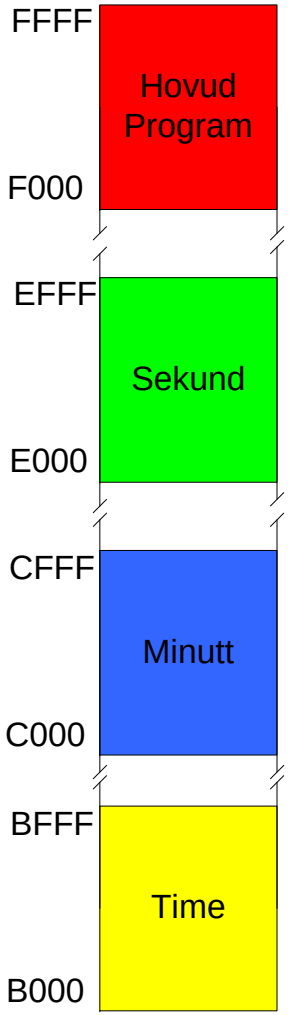
```
Start:    S = 59? (nei) JMP Sekund
          S = 0
          M = 59? (nei) JMP Minutt
          M = 0
          T = 12 (nei) JMP Time
          T = 0
          JMP Start
```

```
Sekund:   S = S + 1
          JMP Start
```

```
Minutt:   M = M + 1
          JMP Start
```

```
Time:     T = T + 1
          JMP Start
```

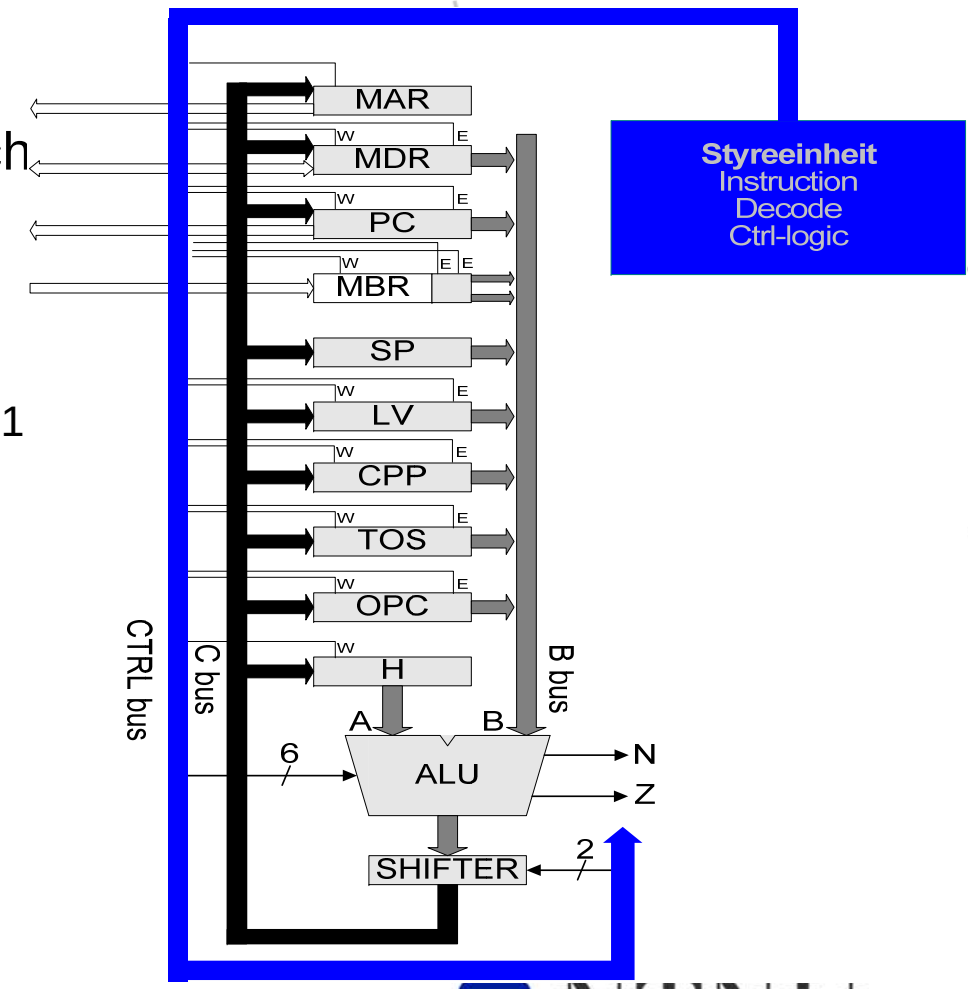
I programminne:



NTNU
Innovation and Creativity

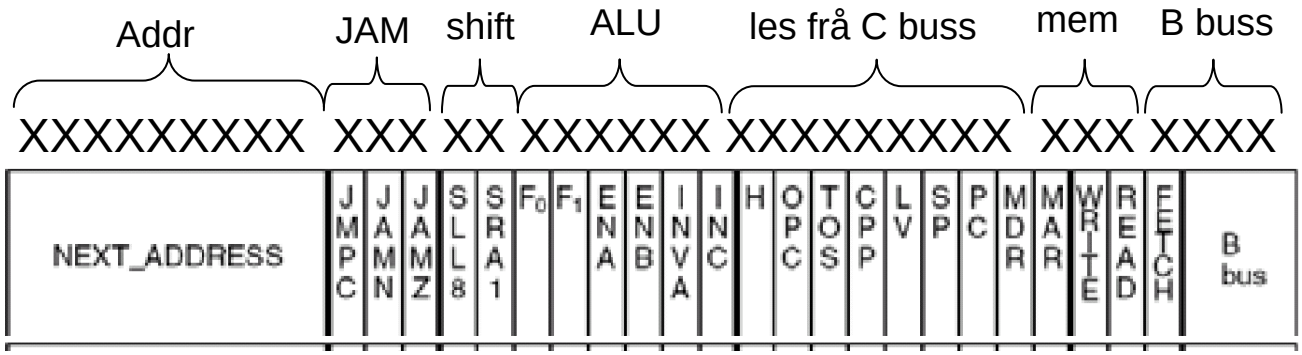
Utføre sekvensar av oprasjonar

- 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)

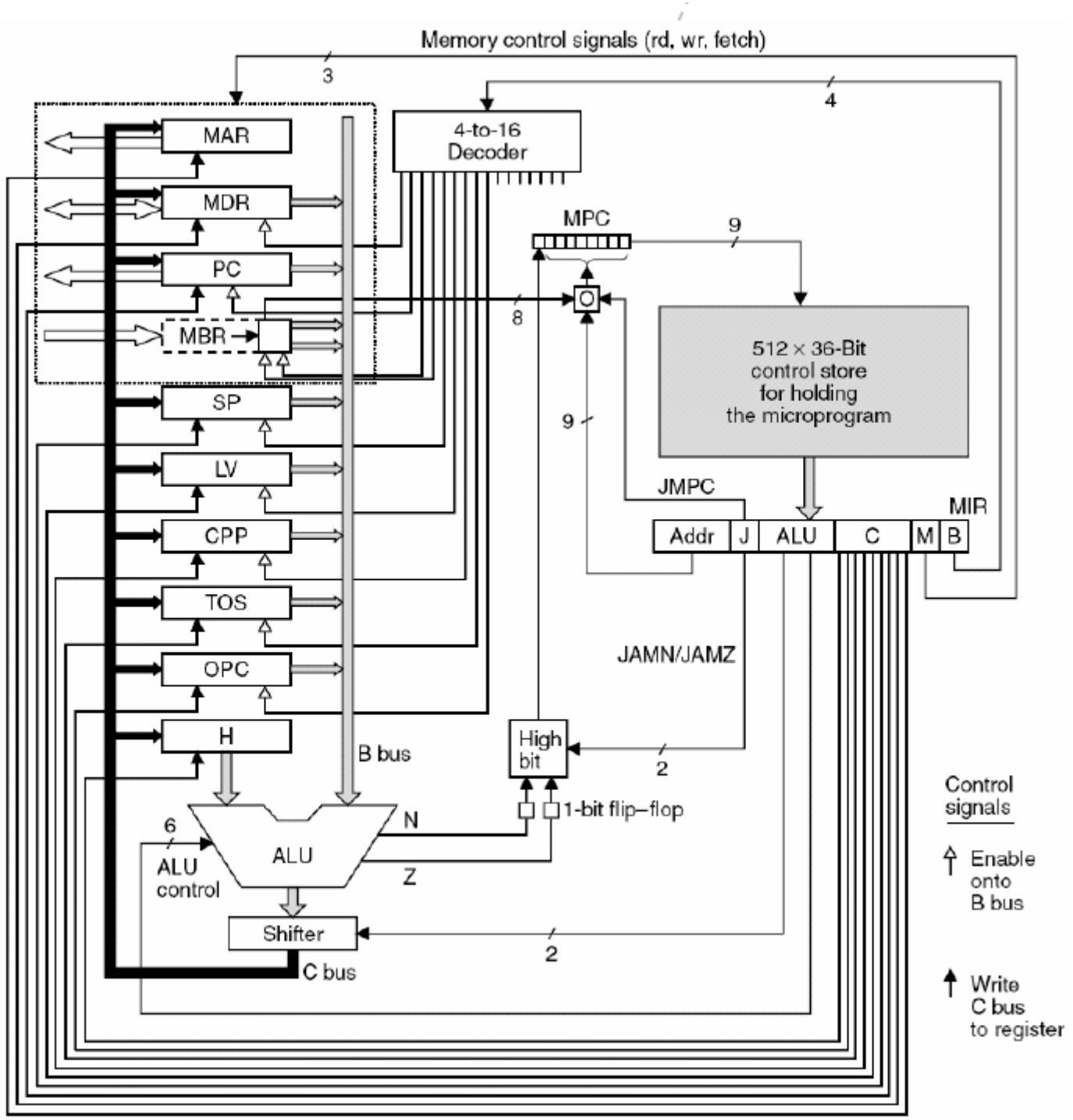


Må ha hopp inn i microinstruksjon

- **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 berekning utført av ALU
  - Kan bruke resultatet til betinga hopp

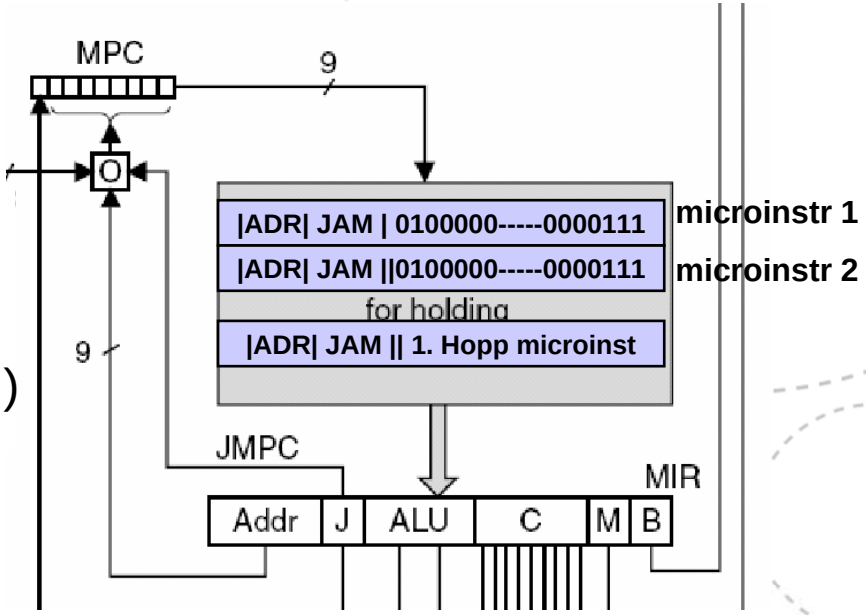


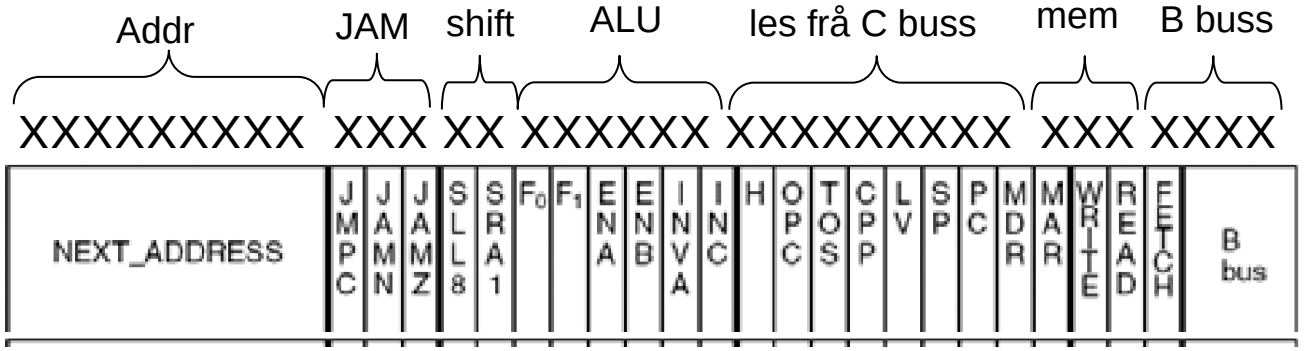
IJVM


Memory control signals (rd, wr, fetch)
3
4
MAR
MDR
PC
MBR
SP
LV
CPP
TOS
OPC
H
4-to-16
Decoder
MPC
9
8
512 × 36-Bit
control store
for holding
the microprogram
9
JMPC
MIR
Addr
J
ALU
C
M
B
JAMN/JAMZ
B bus
High
bit
2
1-bit flip–flop
N
Z
6
ALU
control
ALU
Shifter
2
C bus
Control
signals
Enable
onto
B bus
Write
C bus
to register

Ved hopp

- MPC
  - MicrProgram Counter
  - JAM kan påvirke MPC
- Betingahopp
  - viss (hoppinstr og Z) eller (hoppinstr og N)
    - MPC set til hopp microinstr.
  - Hopp microinstr.
    - microinsstr som utfører eit hopp

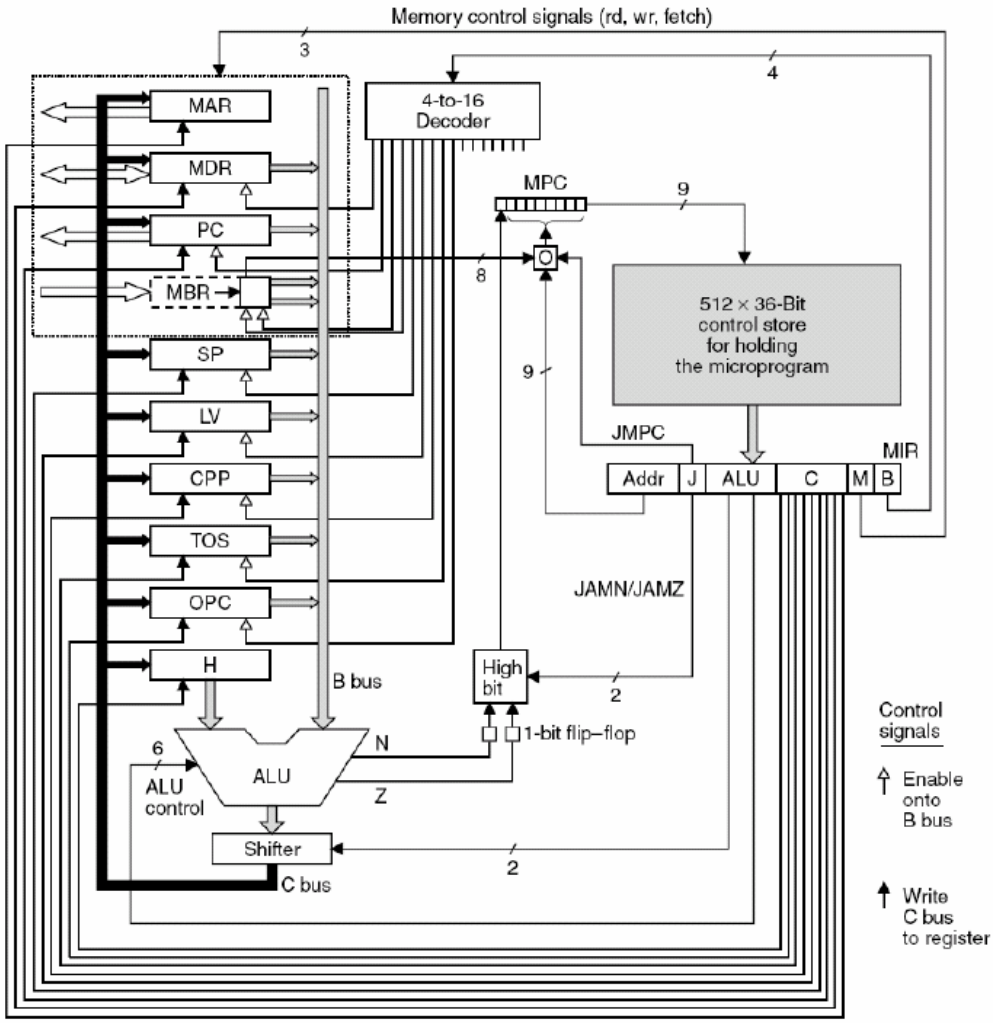




NTNU
Innovation and Creativity

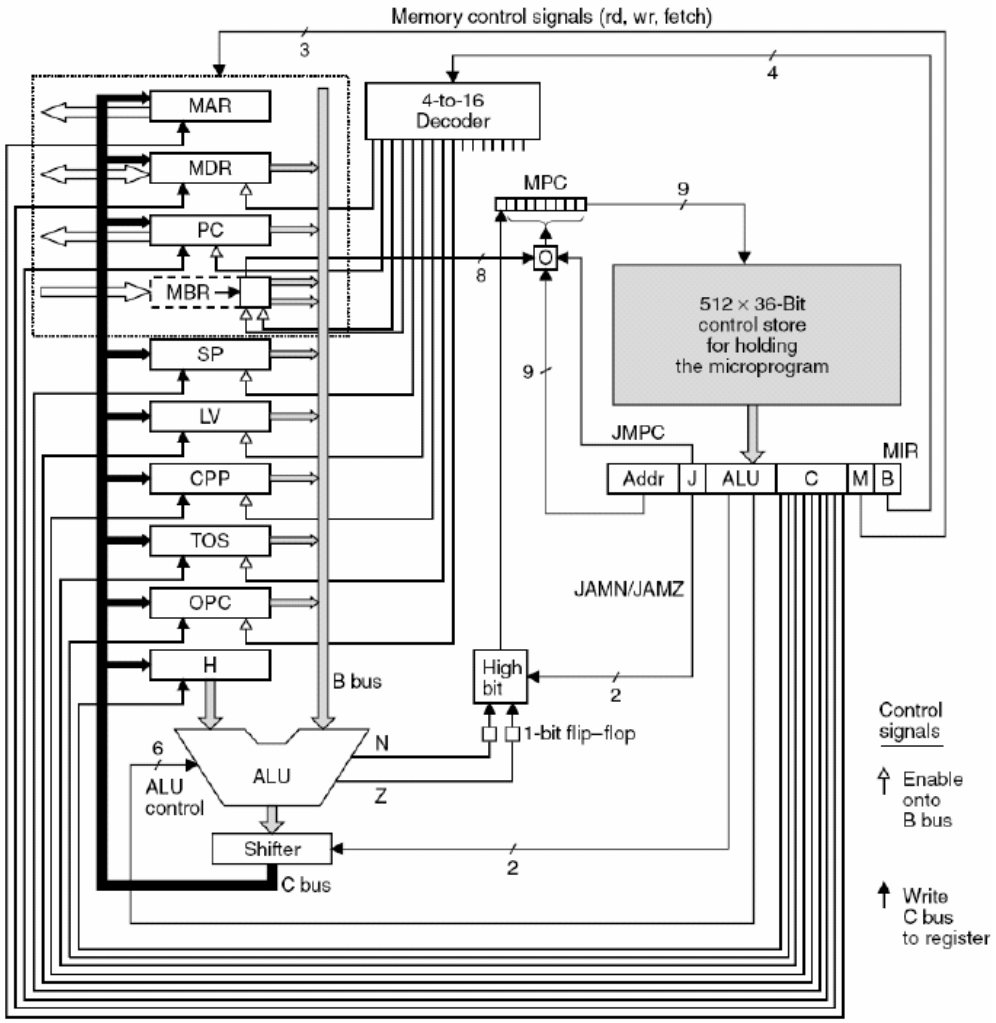
Ved hopp JAMZ og JAMN

- MPC
  - MicrProgram Counter
  - JAMZ og JAMN kan påvirke MPC
- Korleis hopp
  - MPC: 9 bit
    - 9 frå Addr i micro Instruksjon
    - Viss JAMZ eller JAMN =”0”:
      - MPC = Addr
      - Neste microInst Addr
    - Viss JAMZ eller JAMN = ”1”:
      - MPC = Adr or 100000000
      - Neste microInst 1XXXXXXXX



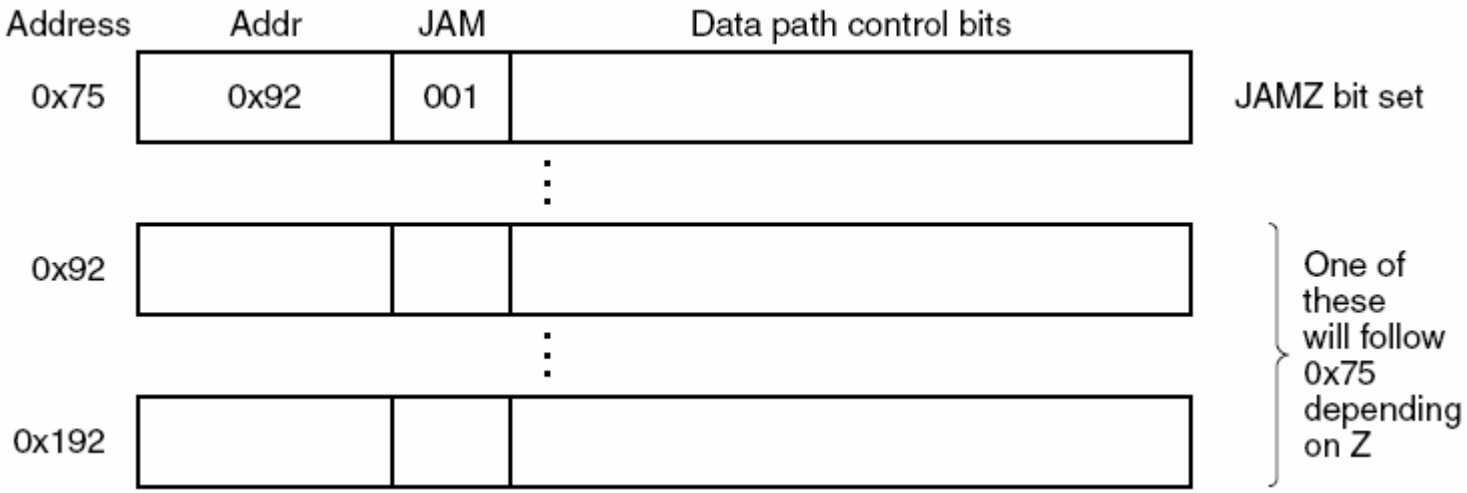
- Ved JAMZ eller JAMN (hopp betingelse i microinstruksjon

  MPC = (Z and JAMZ) or (N and JAMN) or Addr



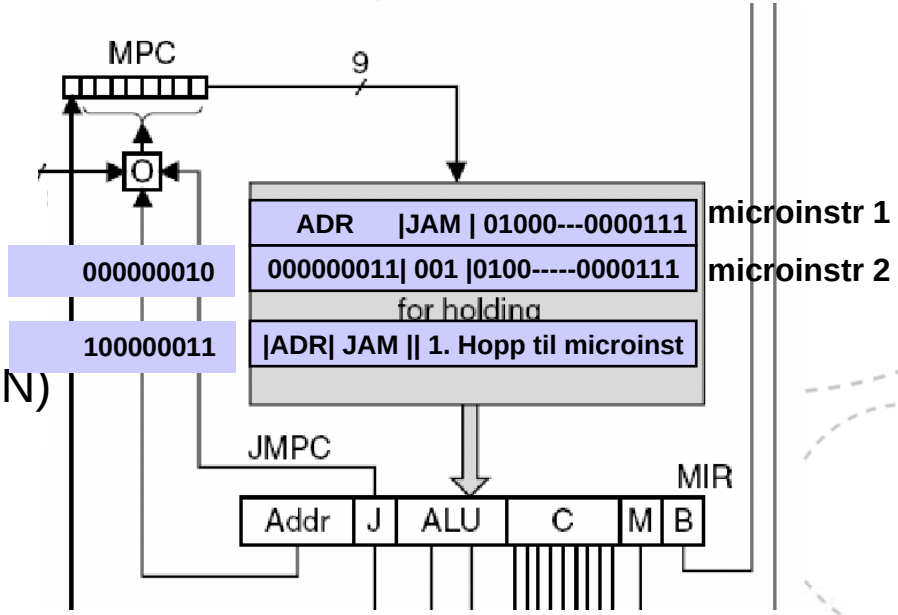
- Ved JAMZ eller JAMN (hopp betingelse i microinstruksjon

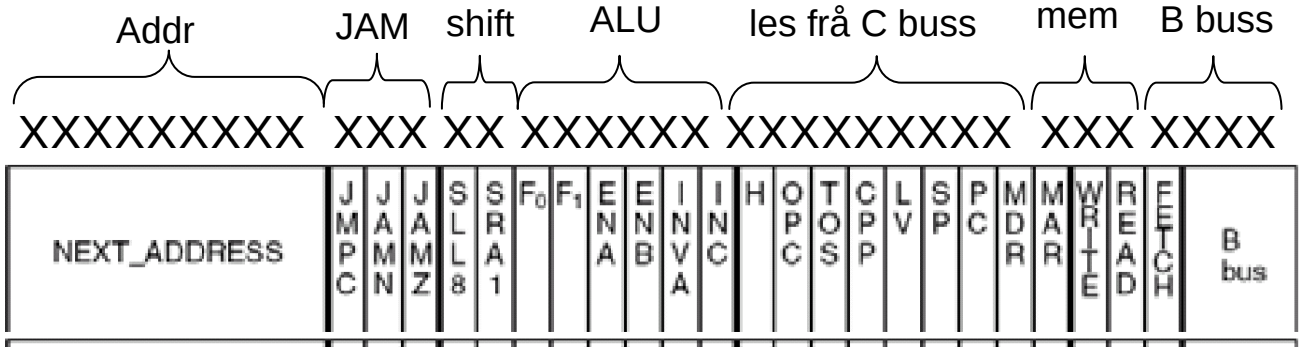
MPC = (Z and JAMN) or (N and JAMN) or Addr



Ved hopp JAMZ og JAMN

- MPC
  - MicrProgram Counter
  - JAM kan påvirke MPC
- Betingahopp
  - viss (hoppinstr og Z) eller (hoppinstr og N)
    - MPC set til hopp microinstr.
  - Hopp microinstr.
    - microinsstr som utfører eit hopp

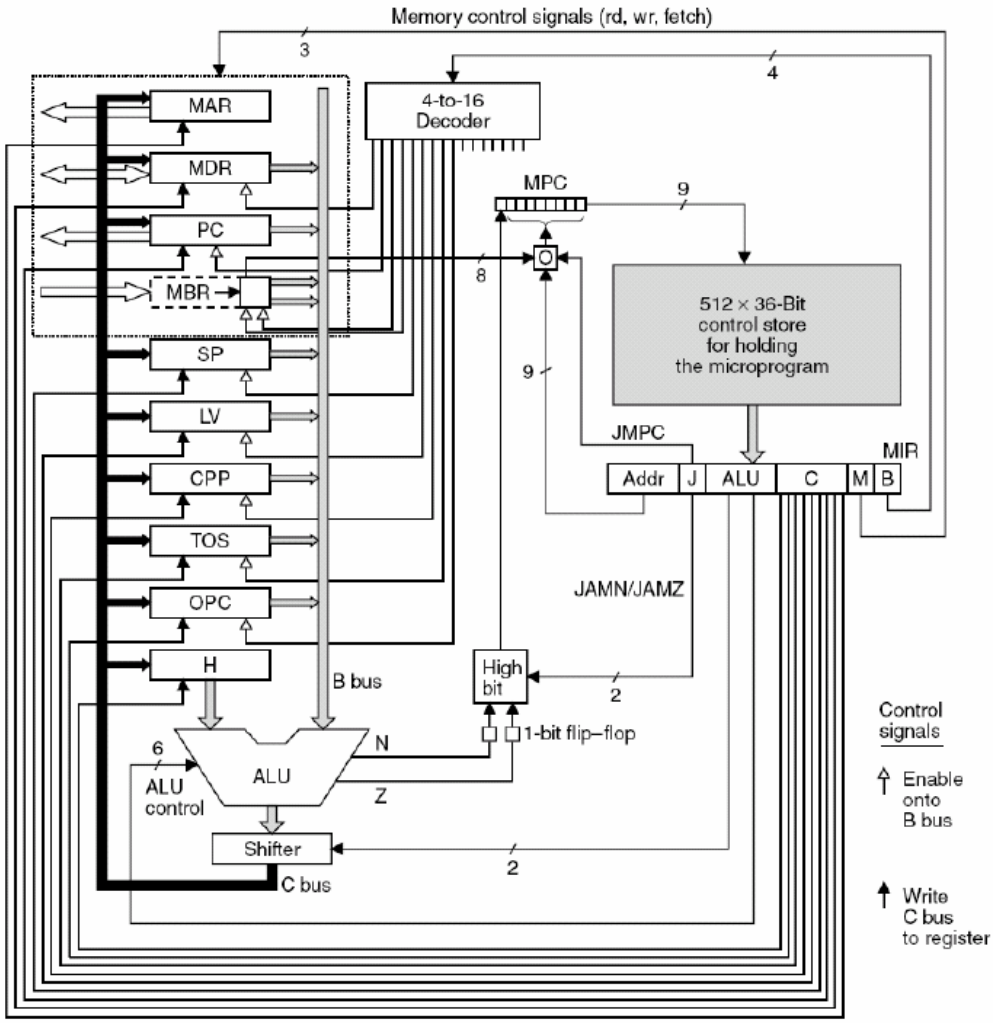




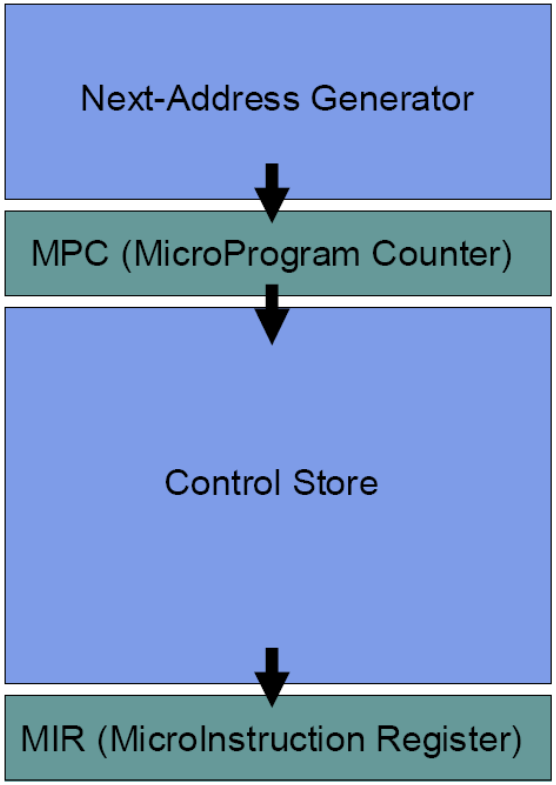
NTNU
Innovation and Creativity

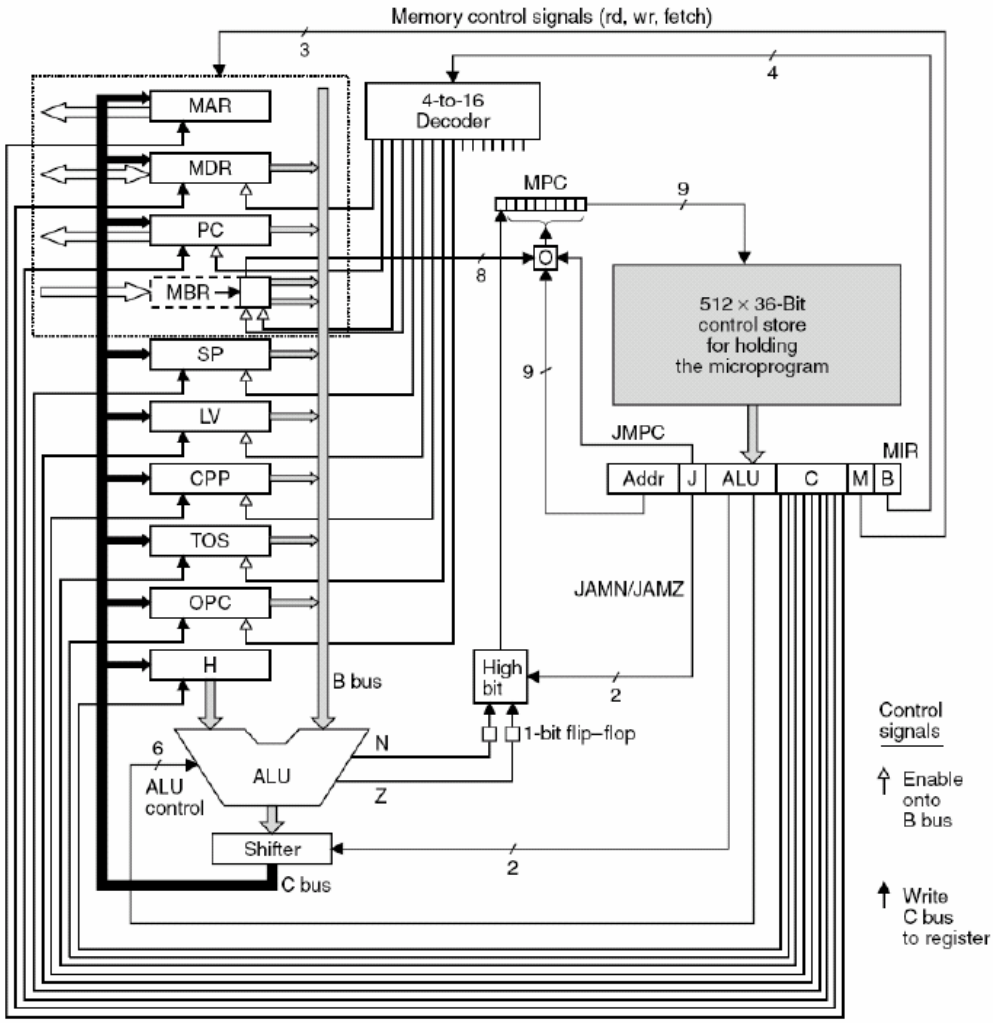
Ved hopp JMPC

- MPC
  - MicrProgram Counter
  - JMPC kan påvirke MPC
- Korleis hopp
  - MPC: 9 bit
    - 9 frå Addr i micro Instruksjon
    - Viss JMPC = ”0”
      - MPC = Addr
      - Neste microInst Addr
    - Viss JMPC = ”1”:
      - MPC = MBR or addr
      - Neste MBR or addr
  - Unik adresse basert på MBR
    - MBR inneheld OpCode

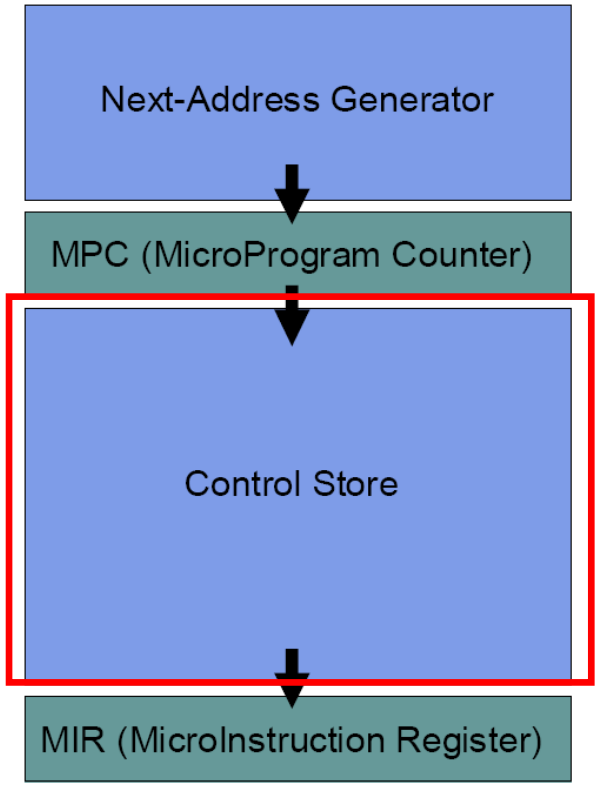


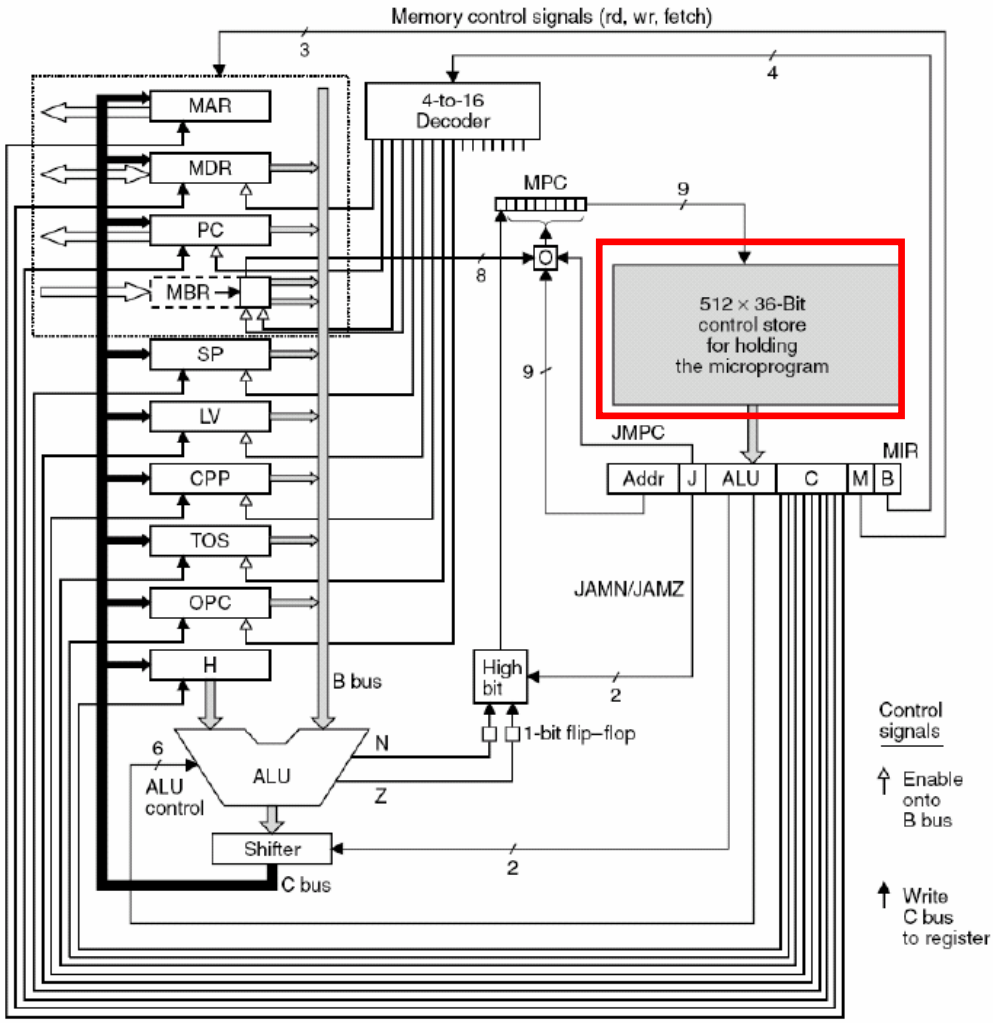
MicroProgramert styreeining IJVM



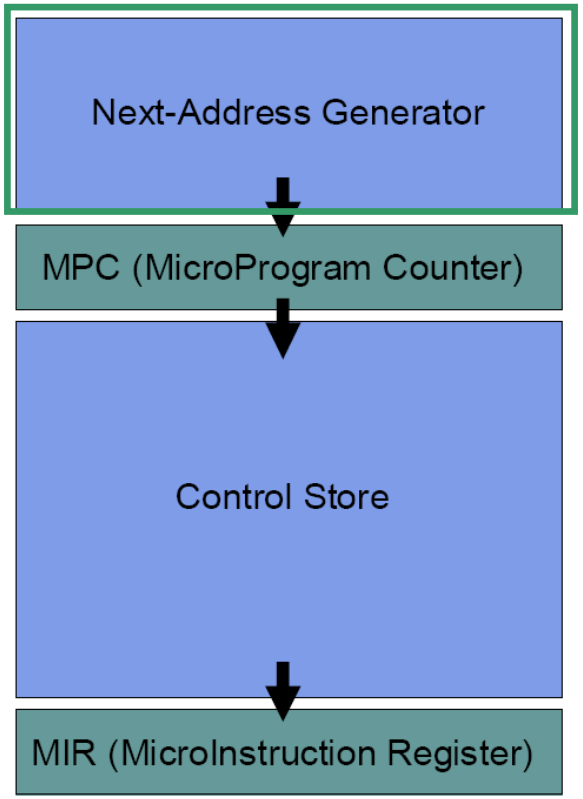


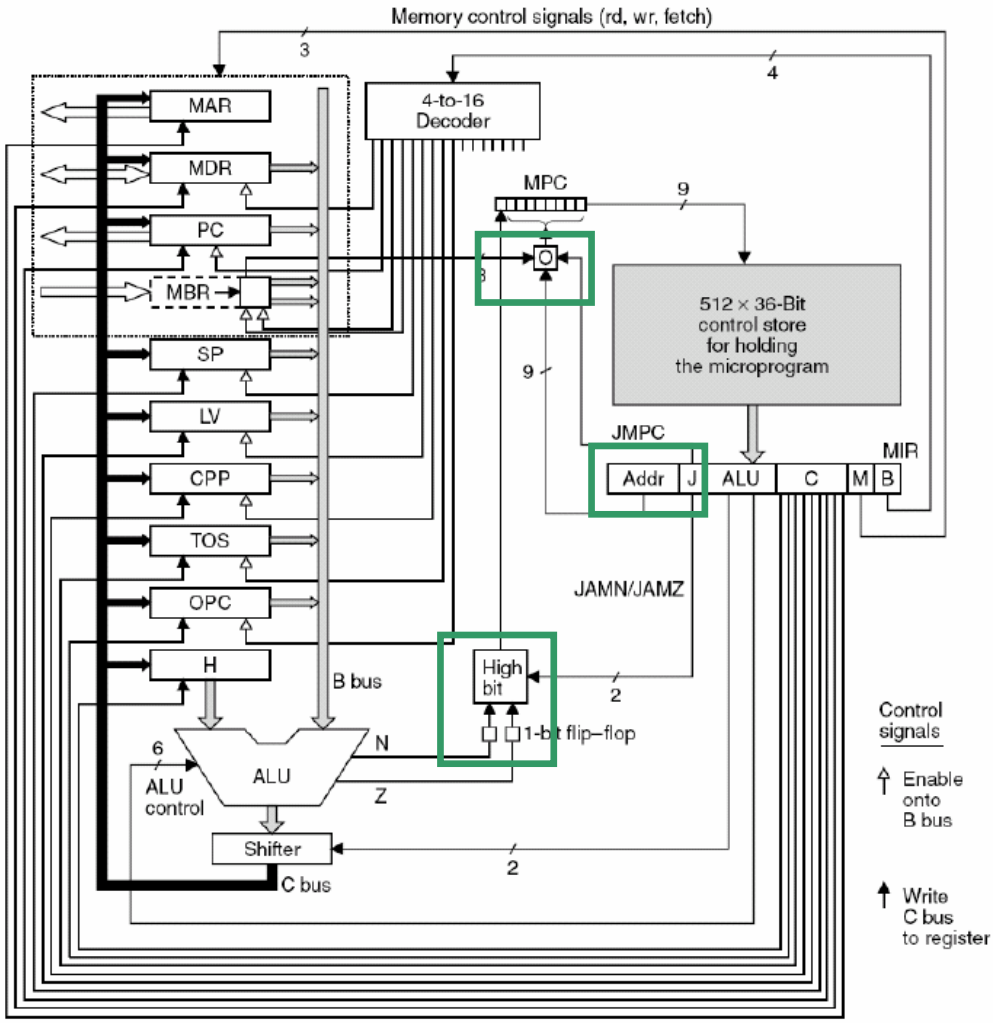
MicroProgramert styreeining IJVM





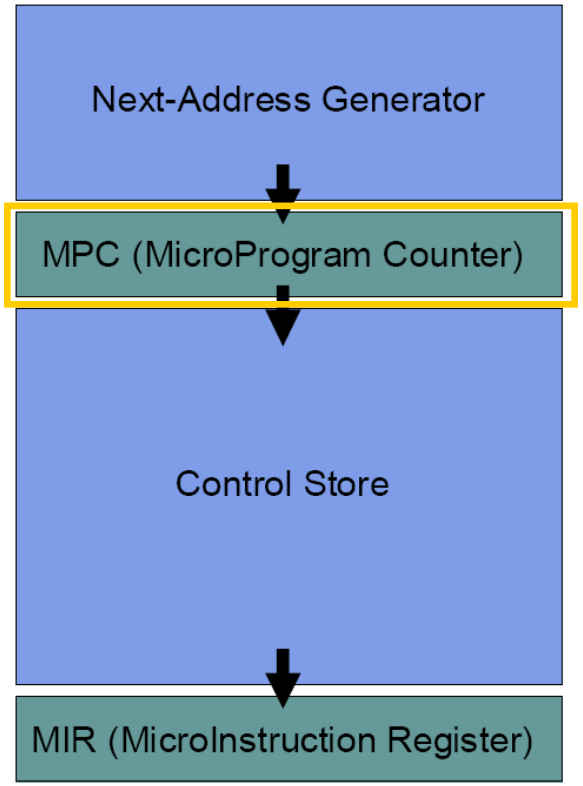
MicroProgramert styreeining IJVM

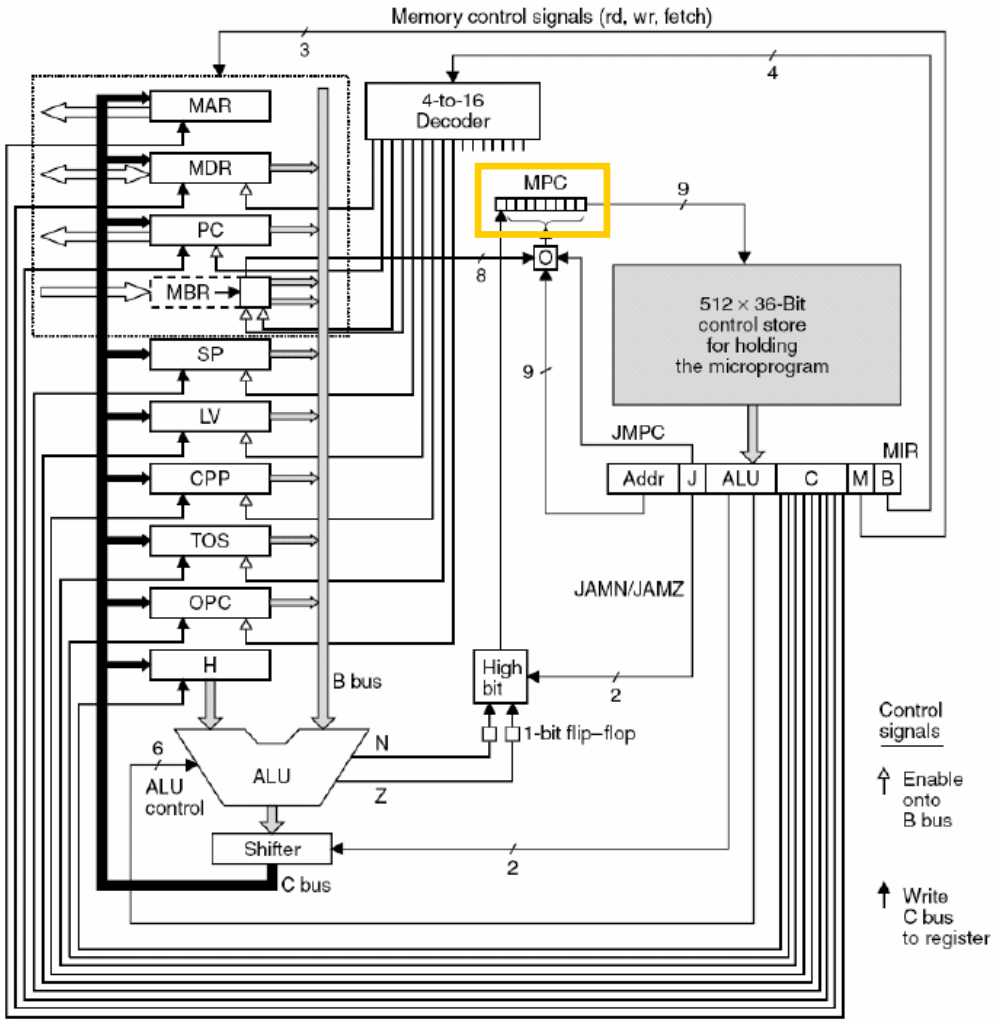




NTNU
Innovation and Creativity

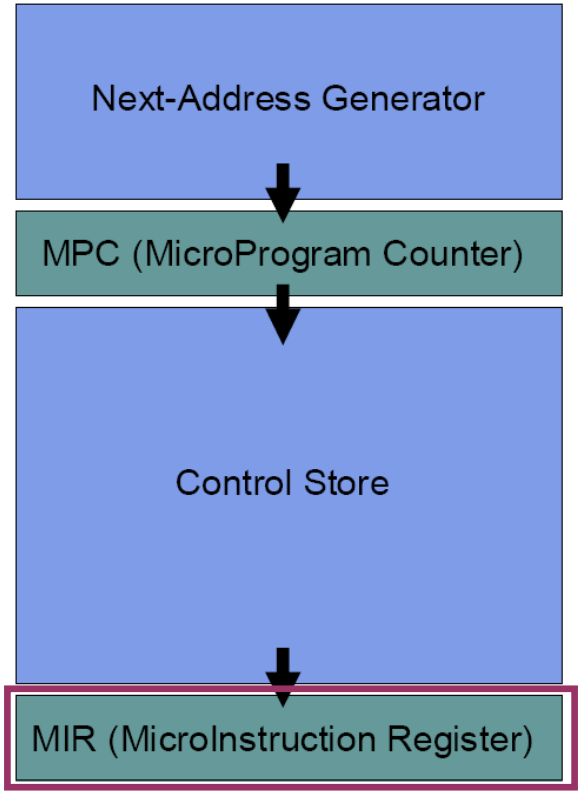
MicroProgramert styreeining IJVM

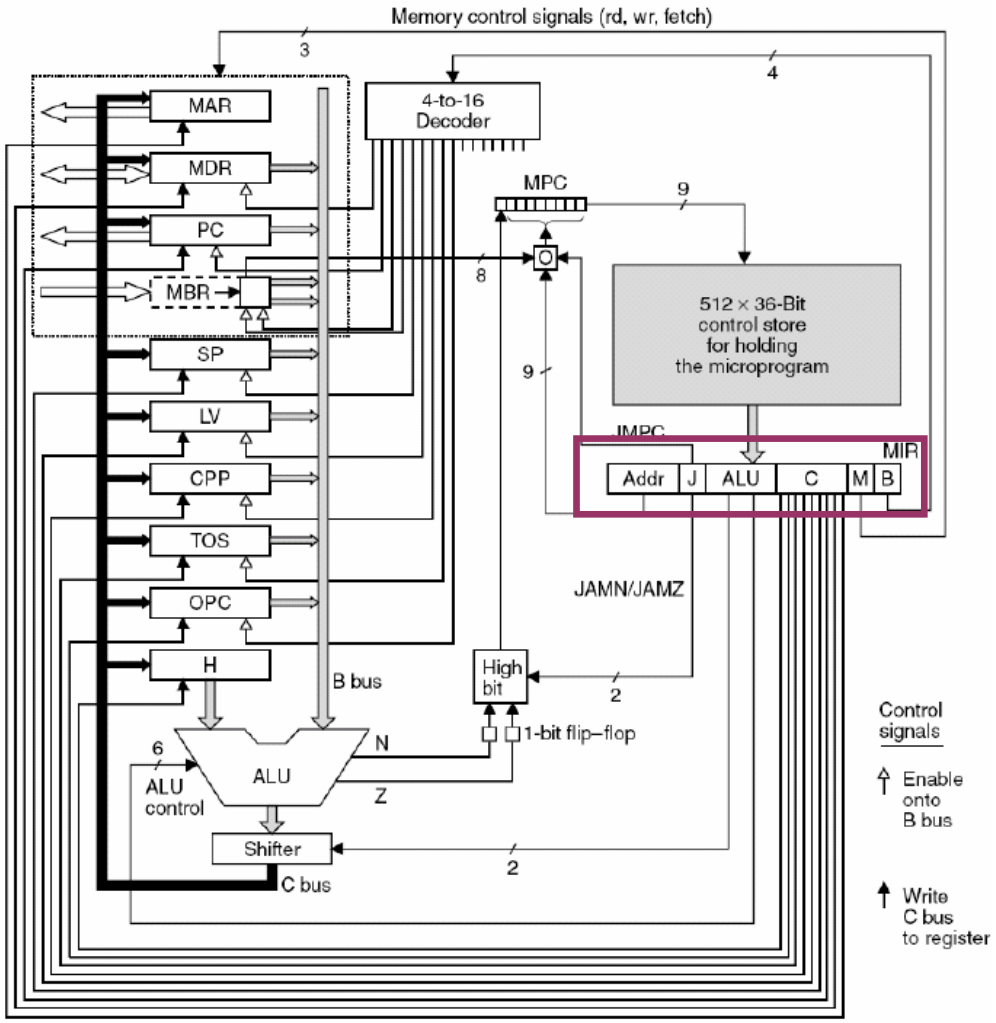




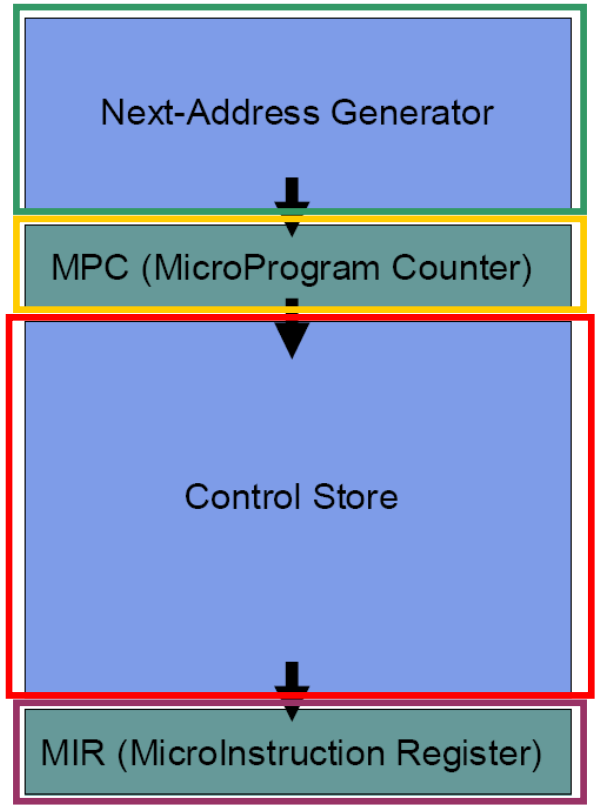
NTNU
Innovation and Creativity

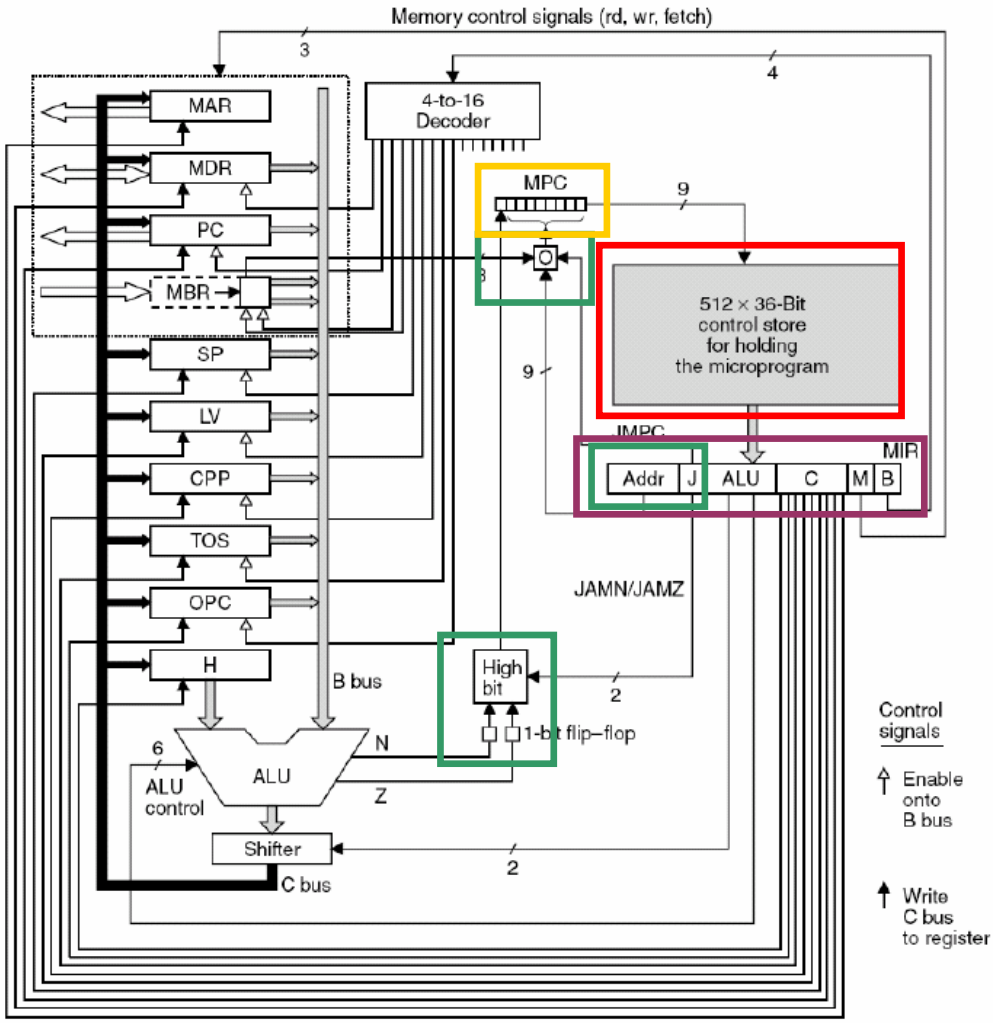
MicroProgramert styreeining IJVM





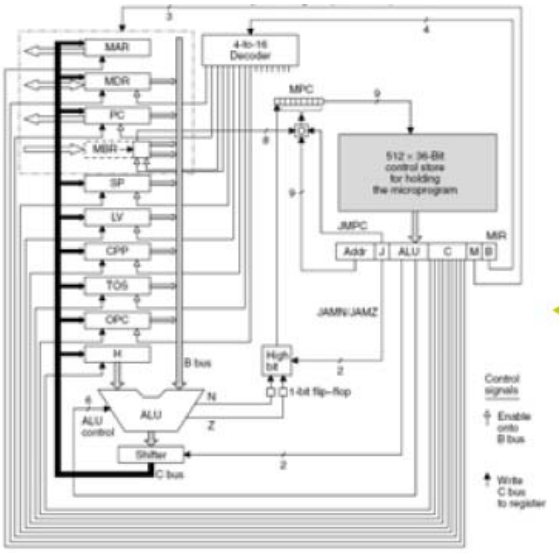
MicroProgramert styreeining IJVM





MicroProgramert styreeining IJVM

- OpCode frå MBR
- Generer adresse til microcode (MPC)
- MPC peikar på 1. microinstruksjon
- Utfør microInst I INSRUKSJONEN

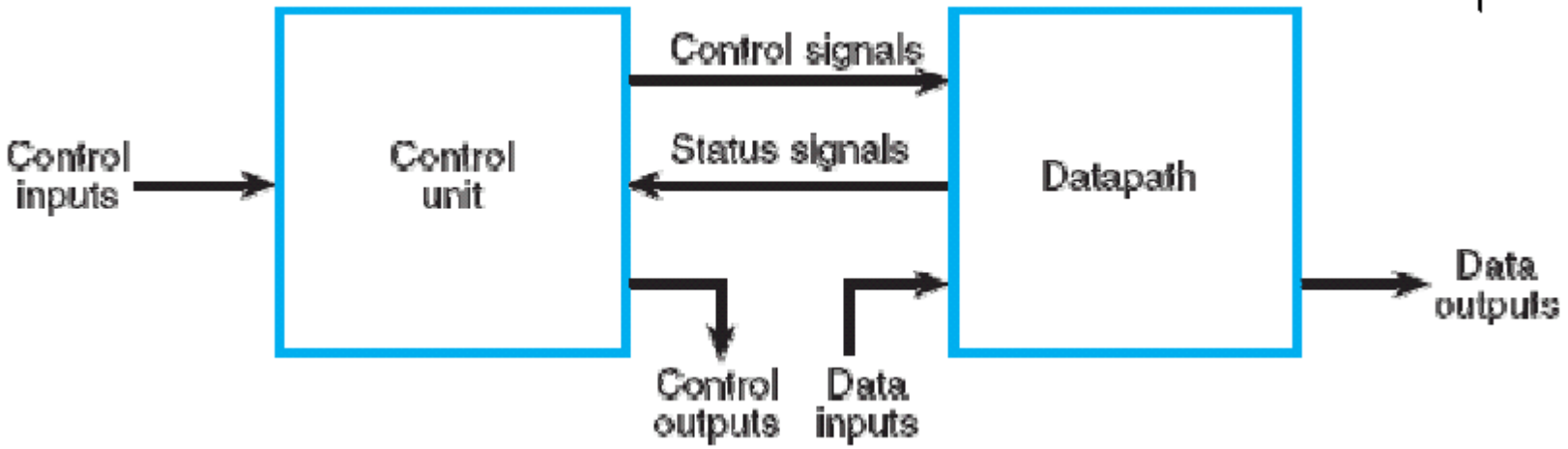


NEXT_ADDRESS	JMPC	JAMN	JAMZ	SLL8	SRA1	F0	F1	ENA	ENB	INVA	INC	H	OPC	TOS	CPP	LV	SP	PC	MDR	MAR	WRITE	READ	FETCH	B bus
NEXT_ADDRESS	JMPC	JAMN	JAMZ	SLL8	SRA1	F0	F1	ENA	ENB	INVA	INC	H	OPC	TOS	CPP	LV	SP	PC	MDR	MAR	WRITE	READ	FETCH	B bus
NEXT_ADDRESS	JMPC	JAMN	JAMZ	SLL8	SRA1	F0	F1	ENA	ENB	INVA	INC	H	OPC	TOS	CPP	LV	SP	PC	MDR	MAR	WRITE	READ	FETCH	B bus

Mikroprogram: Sekvens av μInstr.

NTNU
Innovation and Creativity

Har no ei generell datamaskin

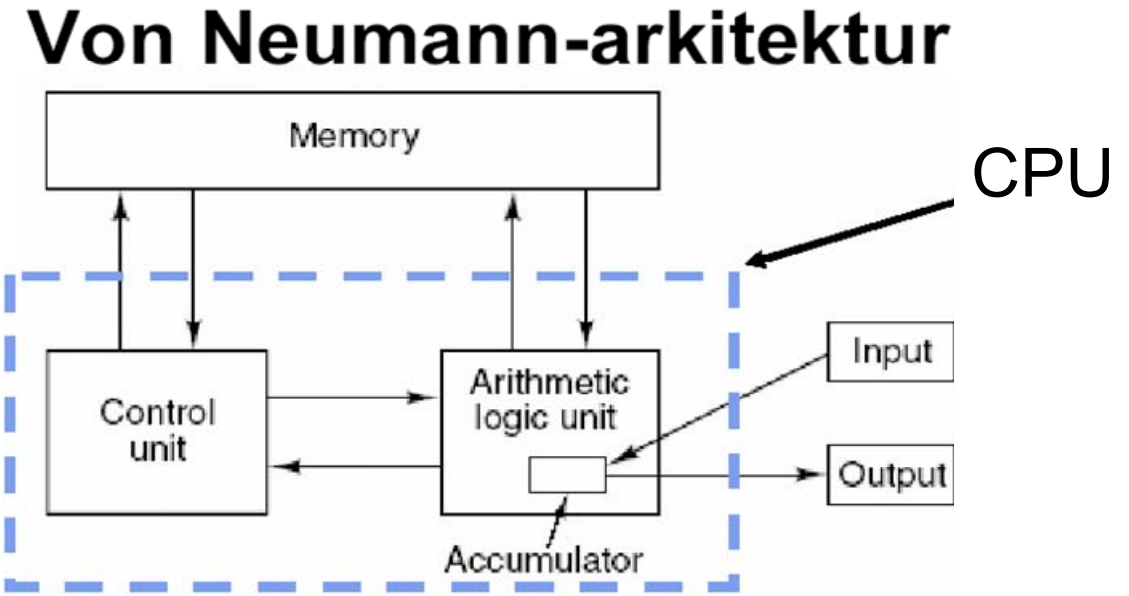


- **Styreeining (control unit)**
- Styrer utførende einingar
  - vha. mikroprogram

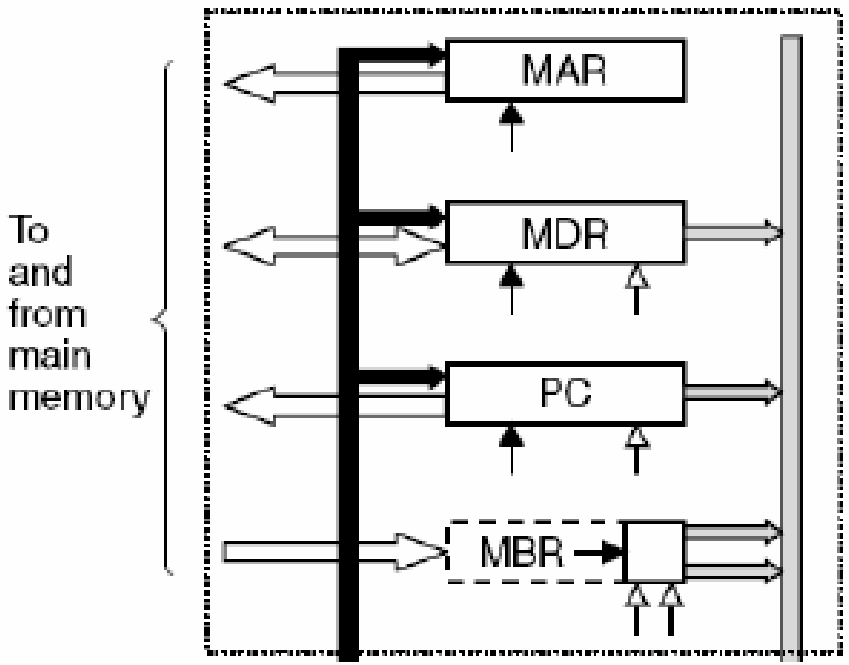
- **Utførende enhet (datapath)**
- Register
- ALU: Aritmetisk-logisk eining

Generell datamaskin minne

- **Minne**
  - Programminne
  - Dataminne



Minne i IJVM

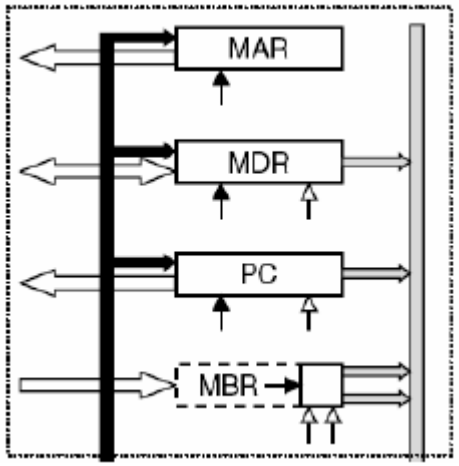


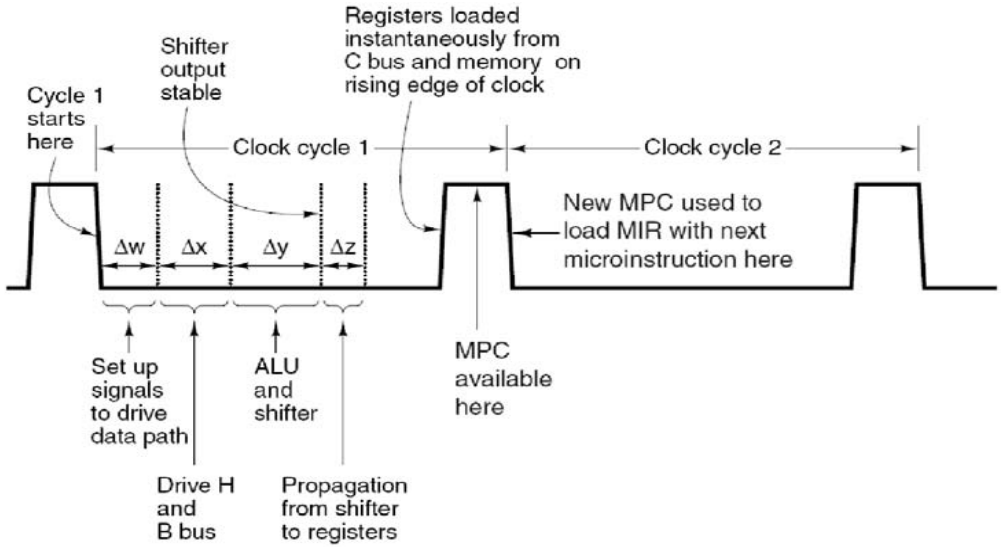
- 32-bit port
  - Memory Address Register
  - Memory Data Register
  - Brukes for data
  - MAR inneholder ordadr.
- 8-bit port
  - Program Counter
  - Memory Buffer Register
  - Brukes for instruksjoner
  - PC inneholder byteadr.
  - 8 bit utvides til 32
    - Enten med 0 eller sign extension

NTNU
Innovation and Creativity

Minne i IJVM

- Klokkesyklus *k*
  - MAR/PC blir oppdatert i slutten av syklus
- Klokkesyklus *k + 1*
  - Antar at minneaksess tar 1 syklus
    - Tilsvarer 100% treffrate på hurtigbuffer (urealistisk)
  - MDR/MBR blir oppdatert i slutten av syklus
- Klokkesyklus *k + 2*
  - Data/Instr. lest fra hovedlager er klart til bruk

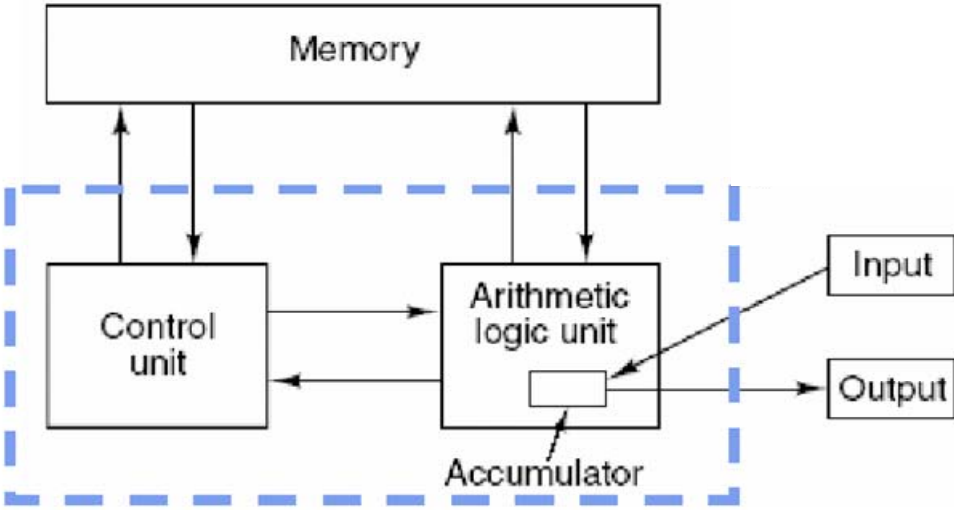


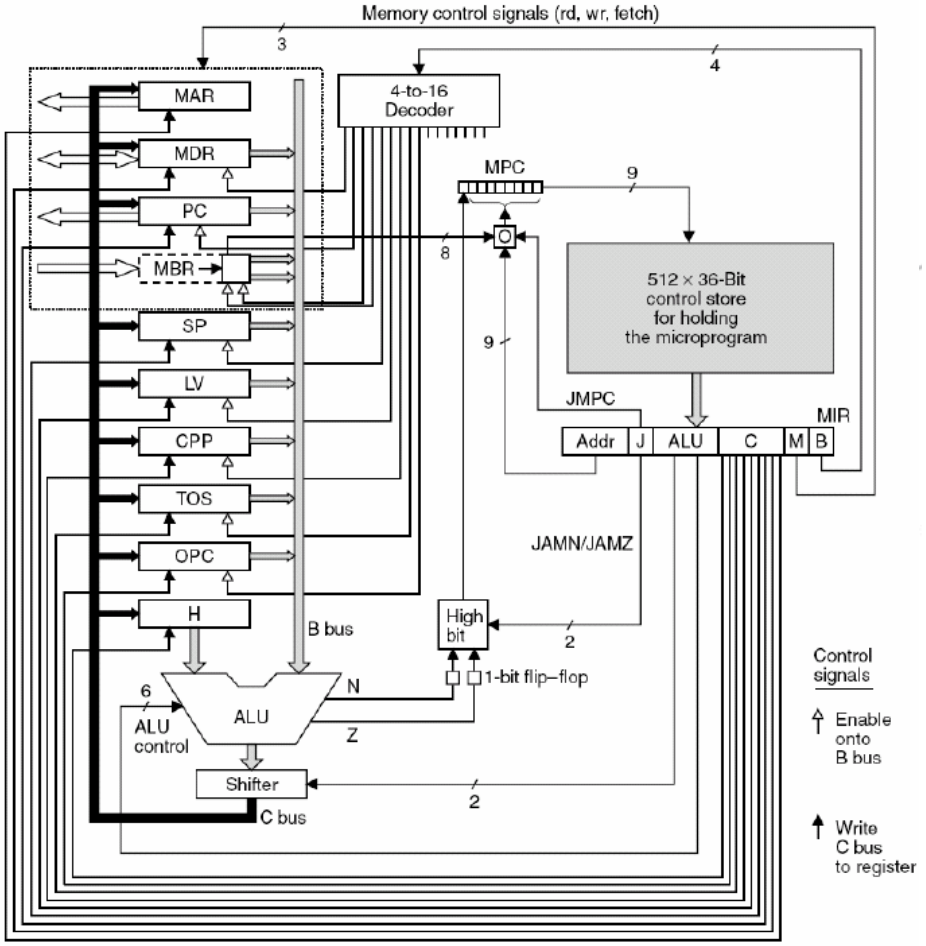


NTNU
Innovation and Creativity

Generell datamaskin

Von Neumann-arkitektur





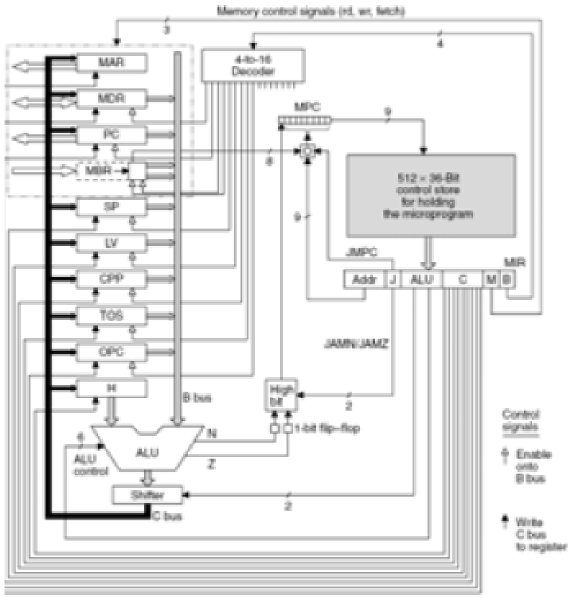
NTNU
Innovation and Creativity

ISA vs. mikroarkitektur

ILOAD j
ILOAD k
IADD
ISTORE i
ILOAD i
BIPUSH 3
...

ISA-instr.

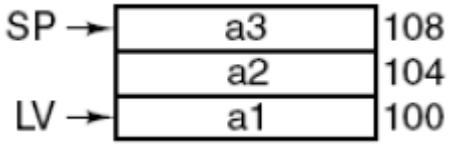
NEXT_ADDRESS	JMPC	JAMN	JAMZ	SLL8	SRA1	F_0	F_1	ENA	ENB	INVA	INC	H	OPC	TOS	CPP	LV	SP	PC	MDR	MAR	WRITE	READ	FETCH	B bus
NEXT_ADDRESS	JMPC	JAMN	JAMZ	SLL8	SRA1	F_0	F_1	ENA	ENB	INVA	INC	H	OPC	TOS	CPP	LV	SP	PC	MDR	MAR	WRITE	READ	FETCH	B bus
NEXT_ADDRESS	JMPC	JAMN	JAMZ	SLL8	SRA1	F_0	F_1	ENA	ENB	INVA	INC	H	OPC	TOS	CPP	LV	SP	PC	MDR	MAR	WRITE	READ	FETCH	B bus



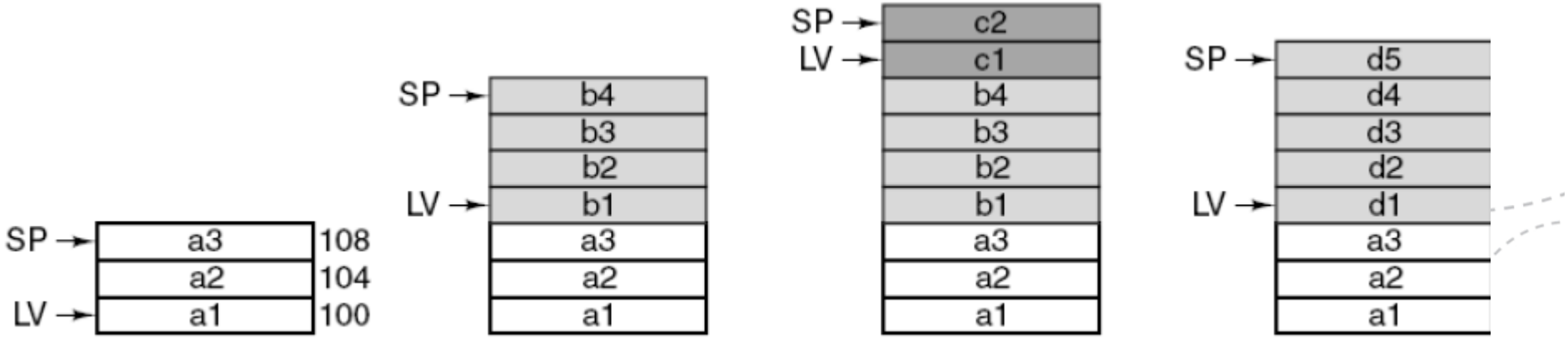
Mikroprogram: Sekvens av μInstr.

IJVM Stakk basert

- Når en metode blir kalt, settes det av plass på stakken til alle lokale variabler
- Register LV (Local variables) peker til der første (nederste) variabel ligger
  - (Register SP (Stack pointer) peker til toppen av stakken)
- Adressen til en lokal variabel blir LV + offset
  - Dermed ikke absolutt minneadresse!
- LV- og SP-adresser er ordadresser



IJVM Stakk basert

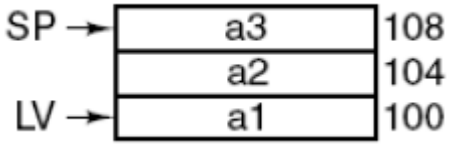


- Figur 1: Metode A har tre lokale variabler
- Figur 2: A kaller B som har fire lokale variabler
- Figur 3: B kaller C som har 2 lokale variabler
- Figur 4: C & B returnerer, A kaller D

NTNU
Innovation and Creativity

IJVM Stakk basert

- Når en metode blir kalt, settes det av plass på stakken til alle lokale variabler
- Register LV (Local variables) peker til der første (nederste) variabel ligger
  - (Register SP (Stack pointer) peker til toppen av stakken)
- Adressen til en lokal variabel blir LV + offset
  - Dermed ikke absolutt minneadresse!
- LV- og SP-adresser er ordadresser

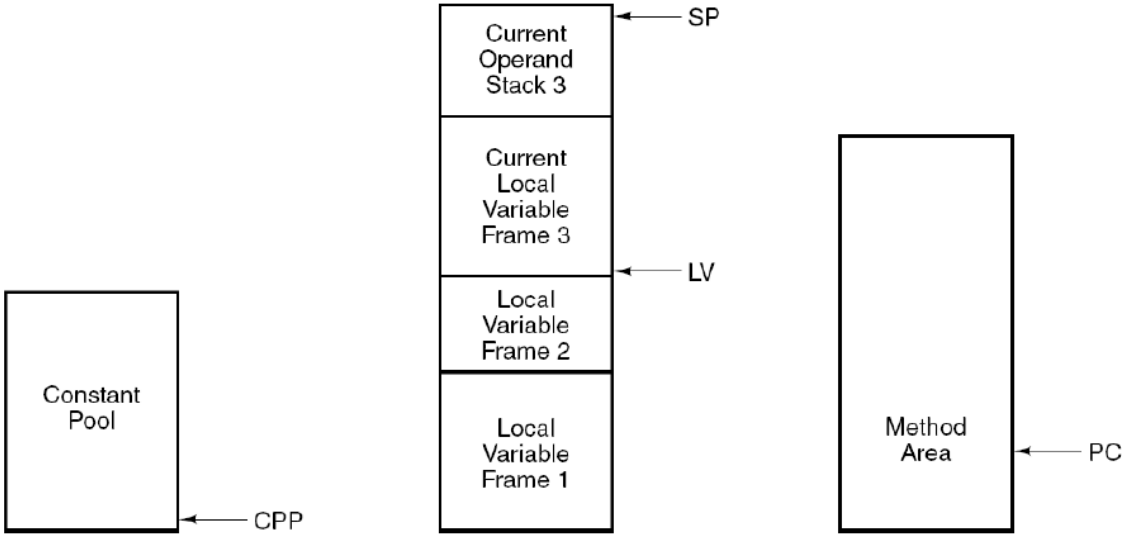


IJVM: MicrArch vs Instruction Set Architecture

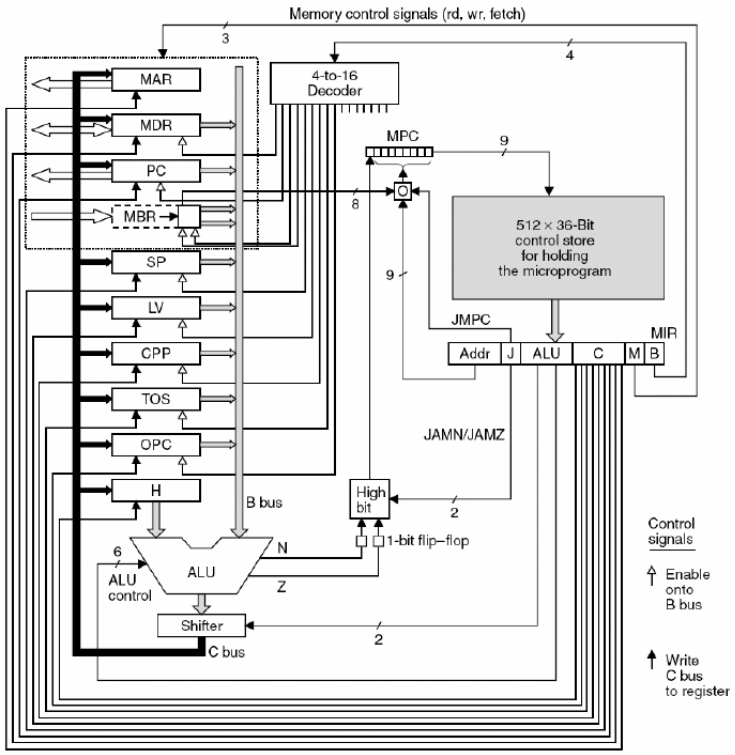
- Instruksjonsset

Hex	Mnemonic	Meaning
0x10	BIPUSH *byte*	Push byte onto stack
0x15	ILOAD *varnum*	Push local variable onto stack
0x13	LDC_W *index*	Push constant from constant pool onto stack
0x57	POP	Delete word on top of stack
0x36	ISTORE *varnum*	Pop word from stack and store in local var.
0x59	DUP	Copy top word on stack and push onto stack
0x5F	SWAP	Swap the two top words on the stack

- Minnemodell



- Micro Arkitektur



NTNU
Innovation and Creativity