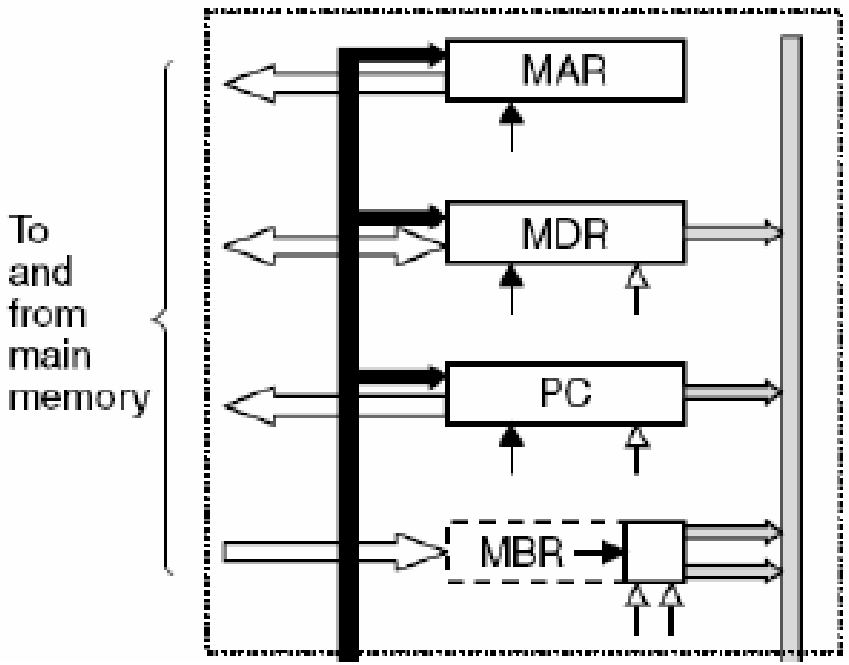
Fortsetelse Microarchitecture level

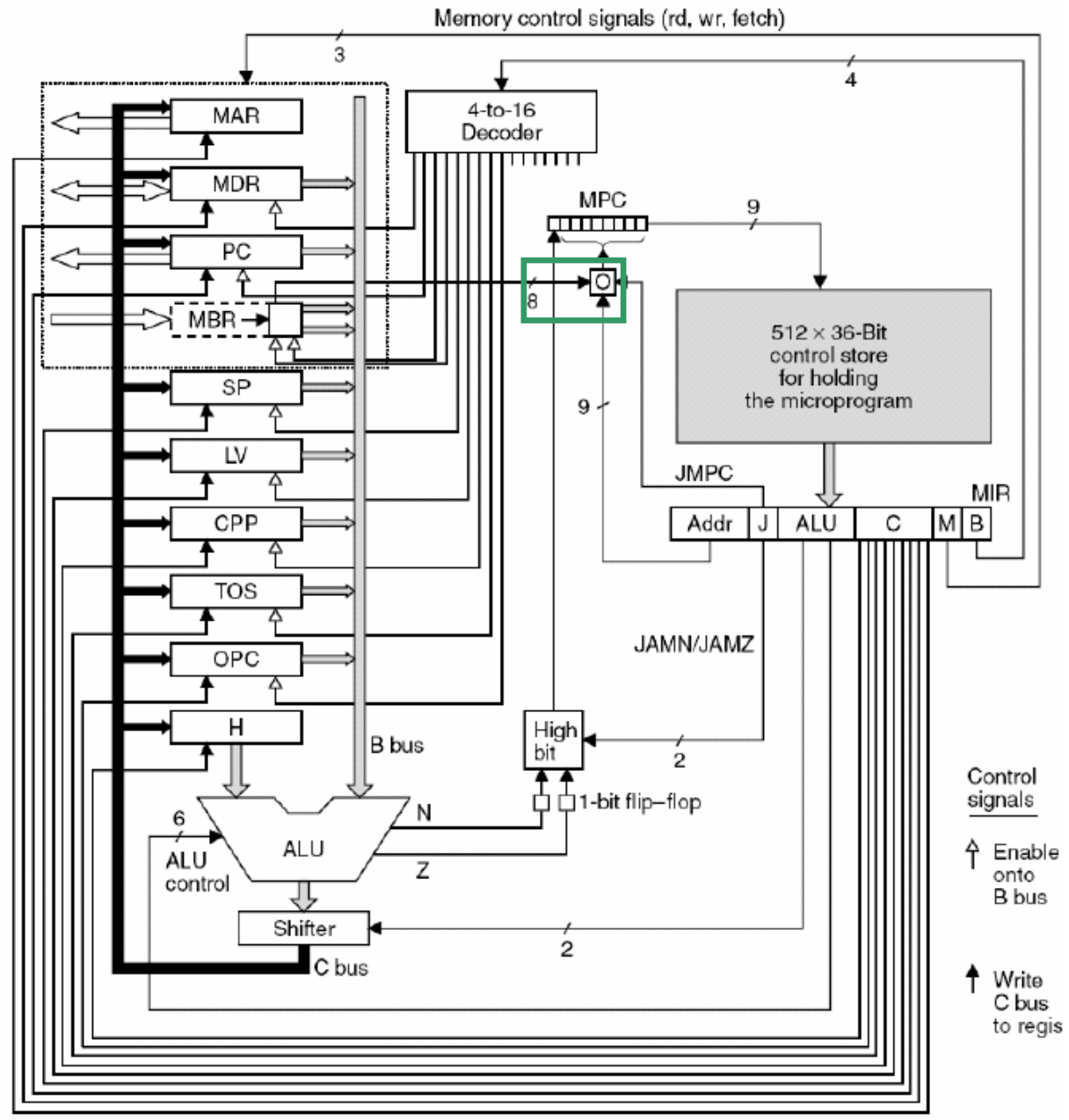
Minne i IJVM



- 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

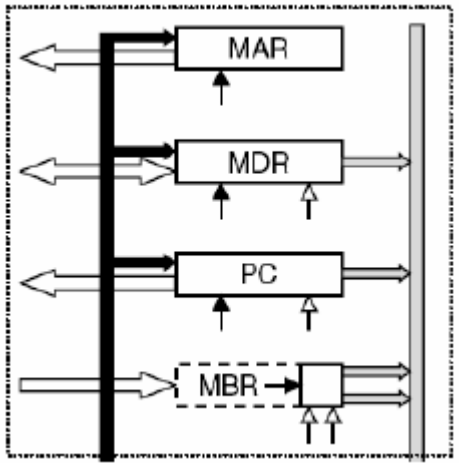
NTNU
Innovation and Creativity

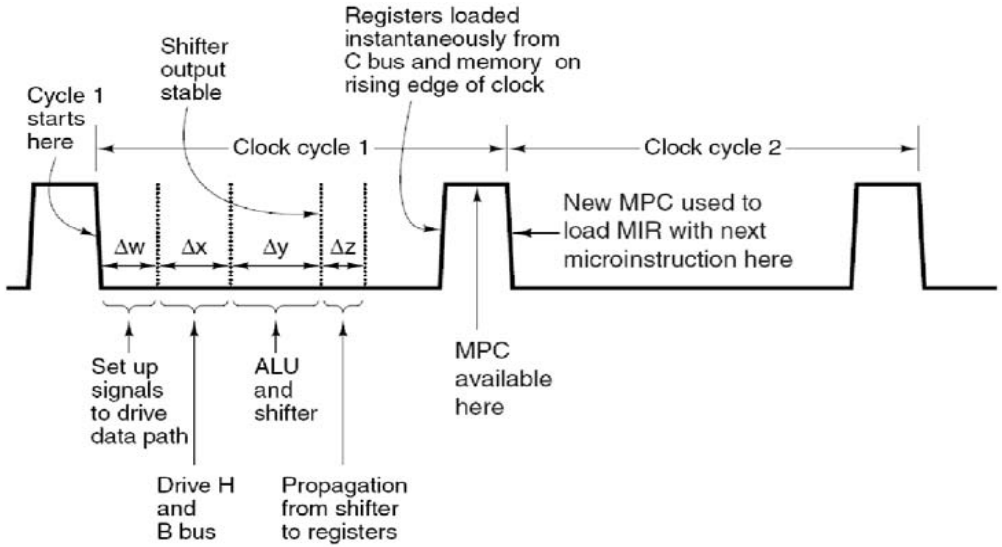
IJVM


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
High
bit
2
B bus
1-bit flip–flop
N
Z
6
ALU
control
ALU
Shifter
2
C bus
Control
signals
Enable
onto
B bus
Write
C bus
to regis

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

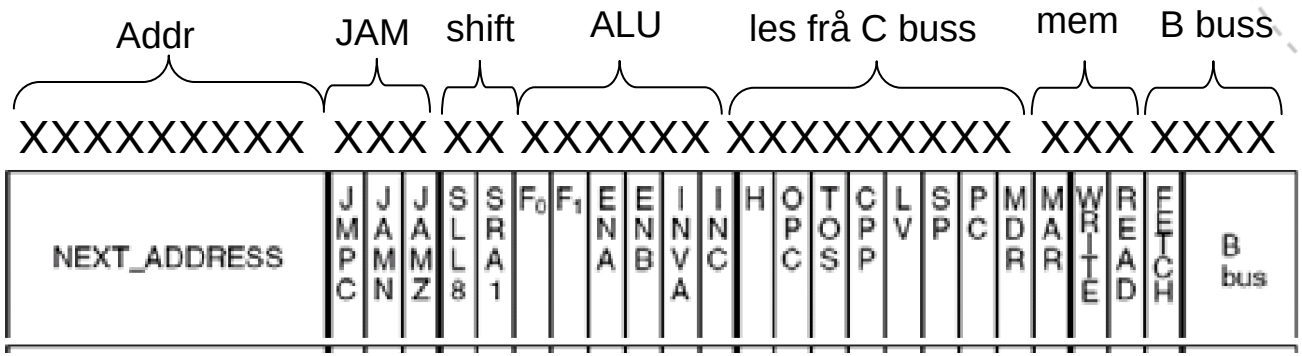




NTNU
Innovation and Creativity

Microinstruksjon mem feltet

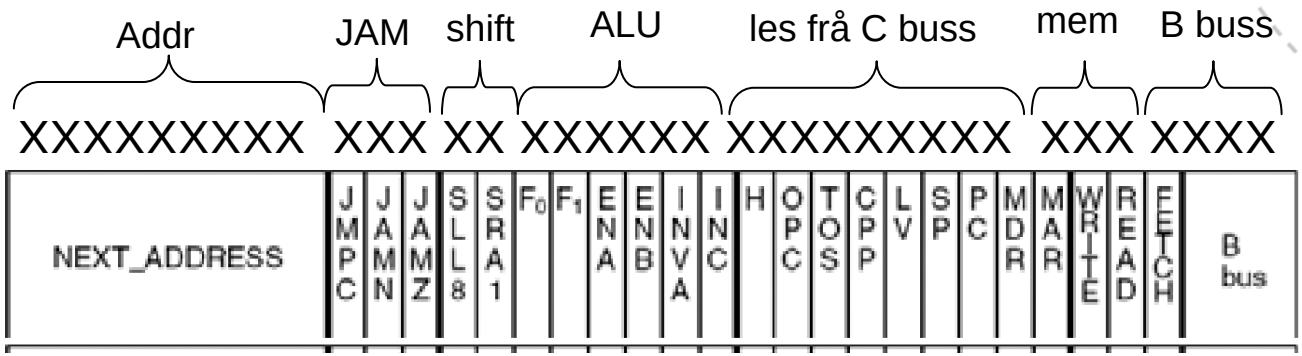
- **Felt for å kontrollere lesing og skriving frå/til minne**



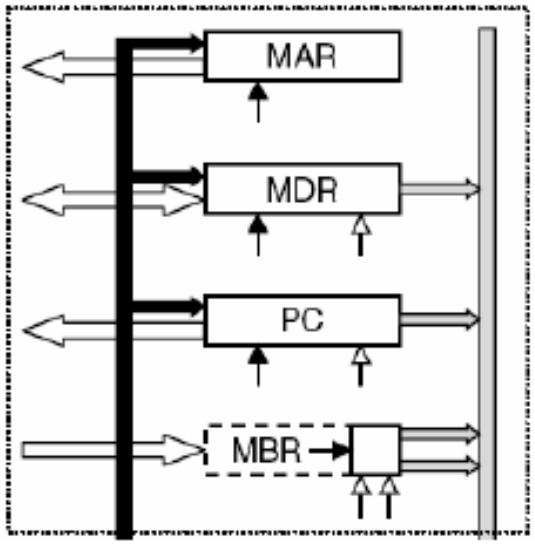
- Minne tilgång
  - Les frå dataminne
  - Skriv til dataminne
  - Hent instruksjonar frå programminne

Microinstruksjon mem feltet

- **Felt for å kontrollere lesing og skriving frå/til minne**

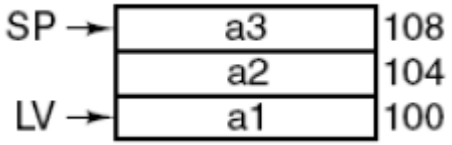


- Minne tilgång
  - Les frå dataminne
    - Read (MAR og MDR)
  - Skriv til dataminne
    - Write (MAR og MDR)
  - Hent instruksjonar frå programminne
    - Fetch (PC og MBR)



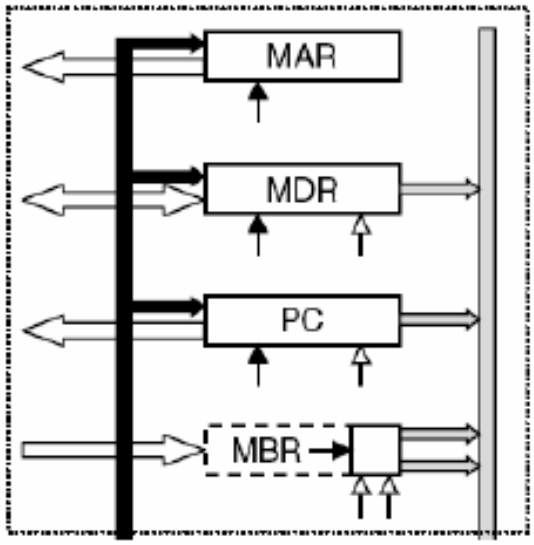
Huks IJVM Stakk basert

- 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



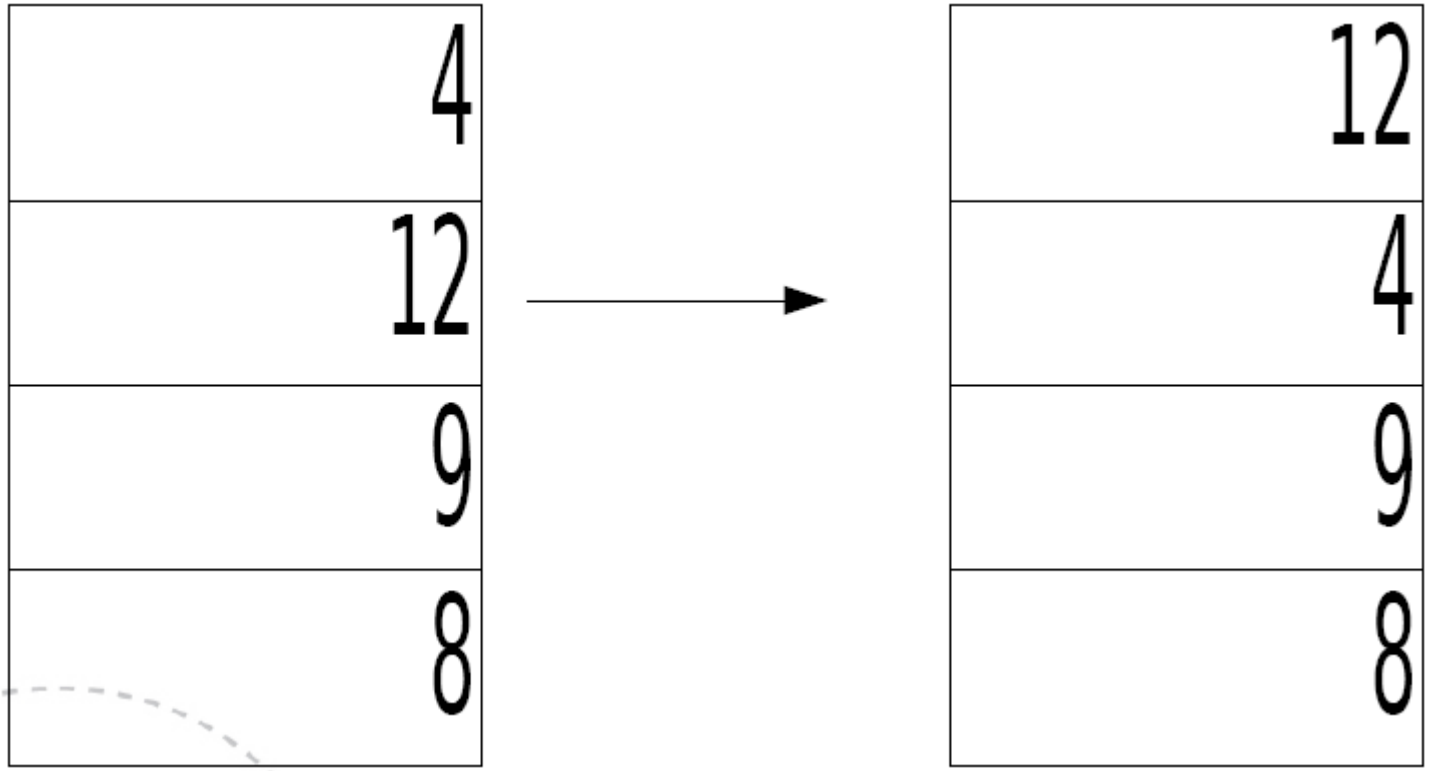
Microinstruksjon mem eksempel

- **Stack operasjonar**
  - **Instruksjon Swap**
  - Byte om på dei to øverste elementa på stacken
  - **Stack i eksternt dataminne**



Microinstruksjon mem eksempel

- **Stack operasjonar**
  - **Instruksjon Swap**
  - Byte om på dei to øverste elementa på stacken
  - **Stack i eksternt dataminne**



NTNU
Innovation and Creativity

Microinstruksjon mem eksempel

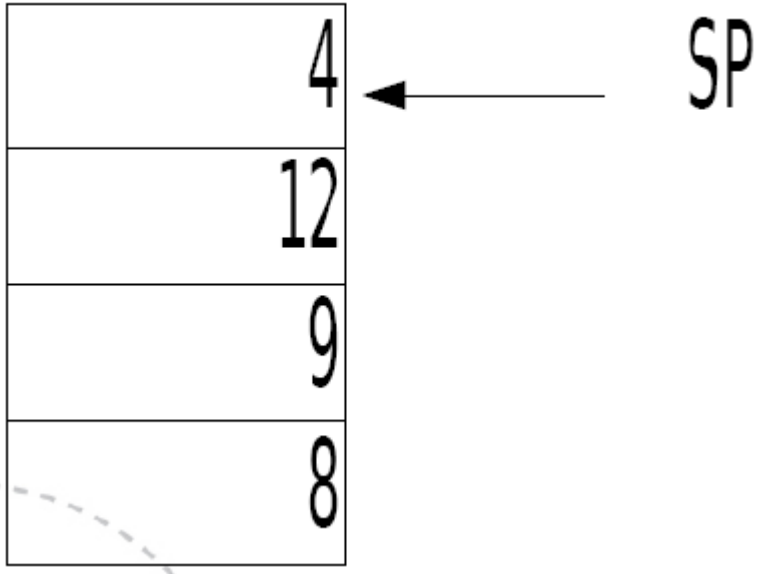
- **Stack operasjonar**
  - **Instruksjon Swap**
  - Byte om på dei to øverste elementa på stacken
  - **Stack i eksternt dataminne**

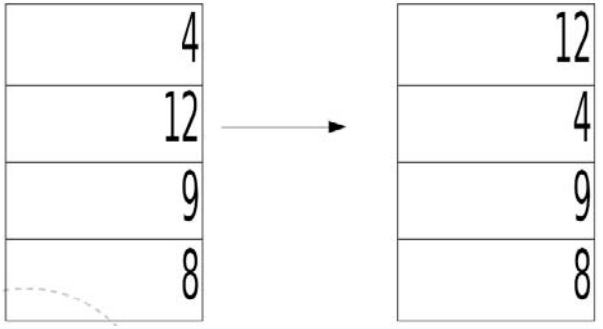
- SP peker på toppen av stakken
- TOS inneholder toppen av stakken
- Main1 : PC = PC + 1; fetch; goto MBR
- MBR = 0x5F
- Swap 1 på adresse 0x5F

NTNU
Innovation and Creativity

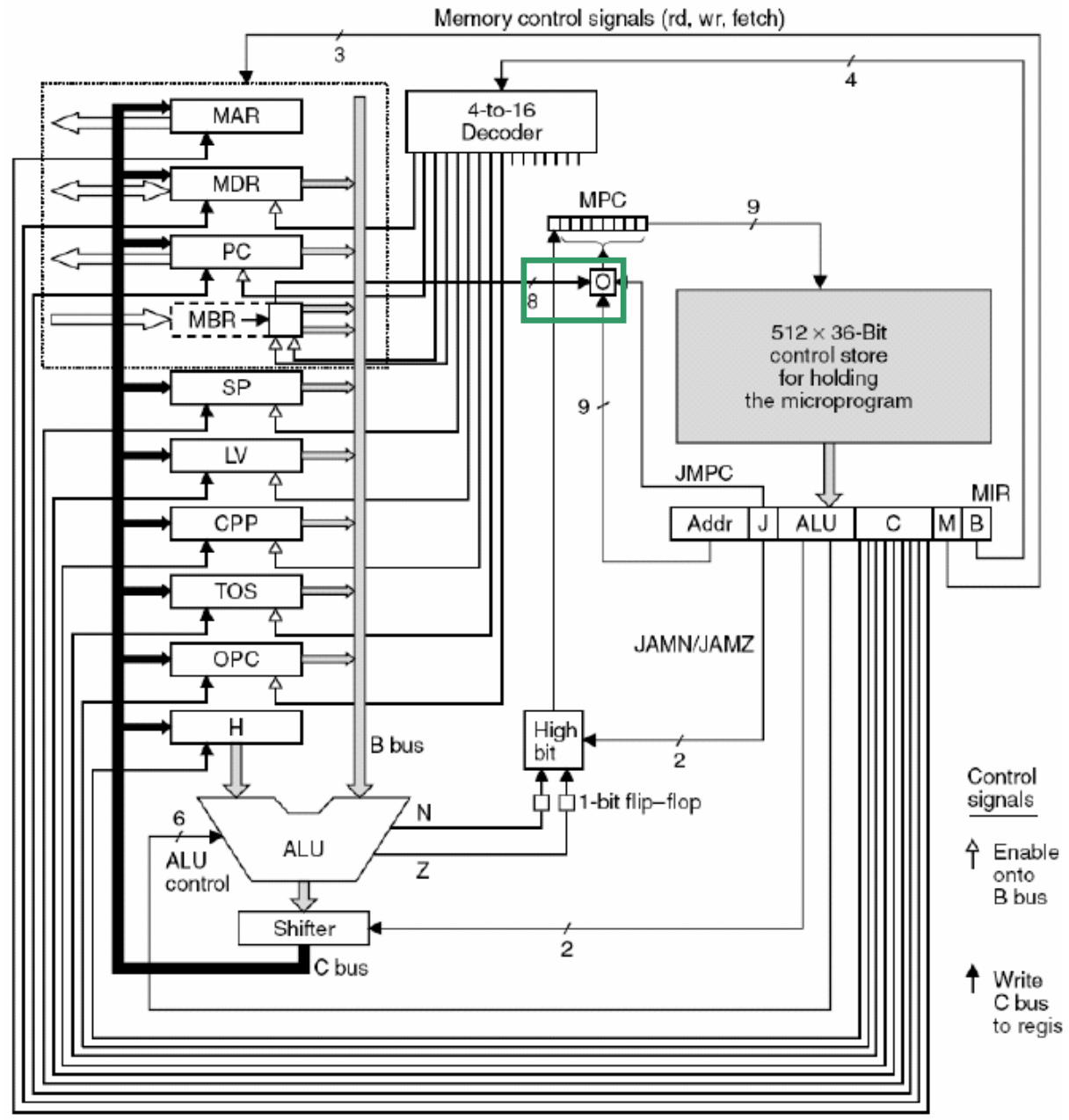
Microinstruksjon mem eksempel

- TOS inneholder 4, trenger å hente inn elementet under
- Skriv andre ordet til toppen av stakken, ta vare på verdien i H
- Skriv den gamle TOS inn i posisjon 2
- Oppdater TOS med ny verdi (fra H)





IJVM

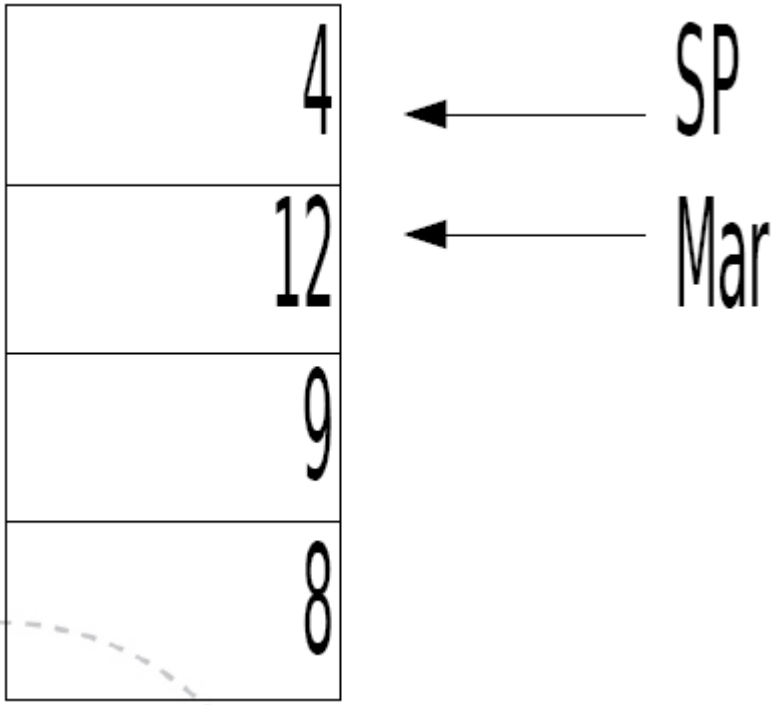

Memory control signals (rd, wr, fetch)
3
4
MAR
MDR
PC
MBR
SP
LV
CPP
TOS
OPC
H
4-to-16
Decoder
MPC
9
8
512 × 36-Bit
control store
for holding
the microprogram
9
JMPC
MIR
Addr
J
ALU
C
M
B
JAMN/JAMZ
B bus
High
bit
2
1-bit flip–flop
N
Z
6
ALU
control
ALU
Shifter
2
C bus
Control
signals
Enable
onto
B bus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

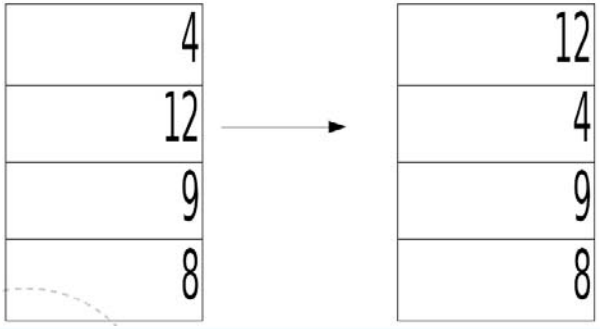
Microinstruksjon mem eksempel

Les inn ord nummer to:

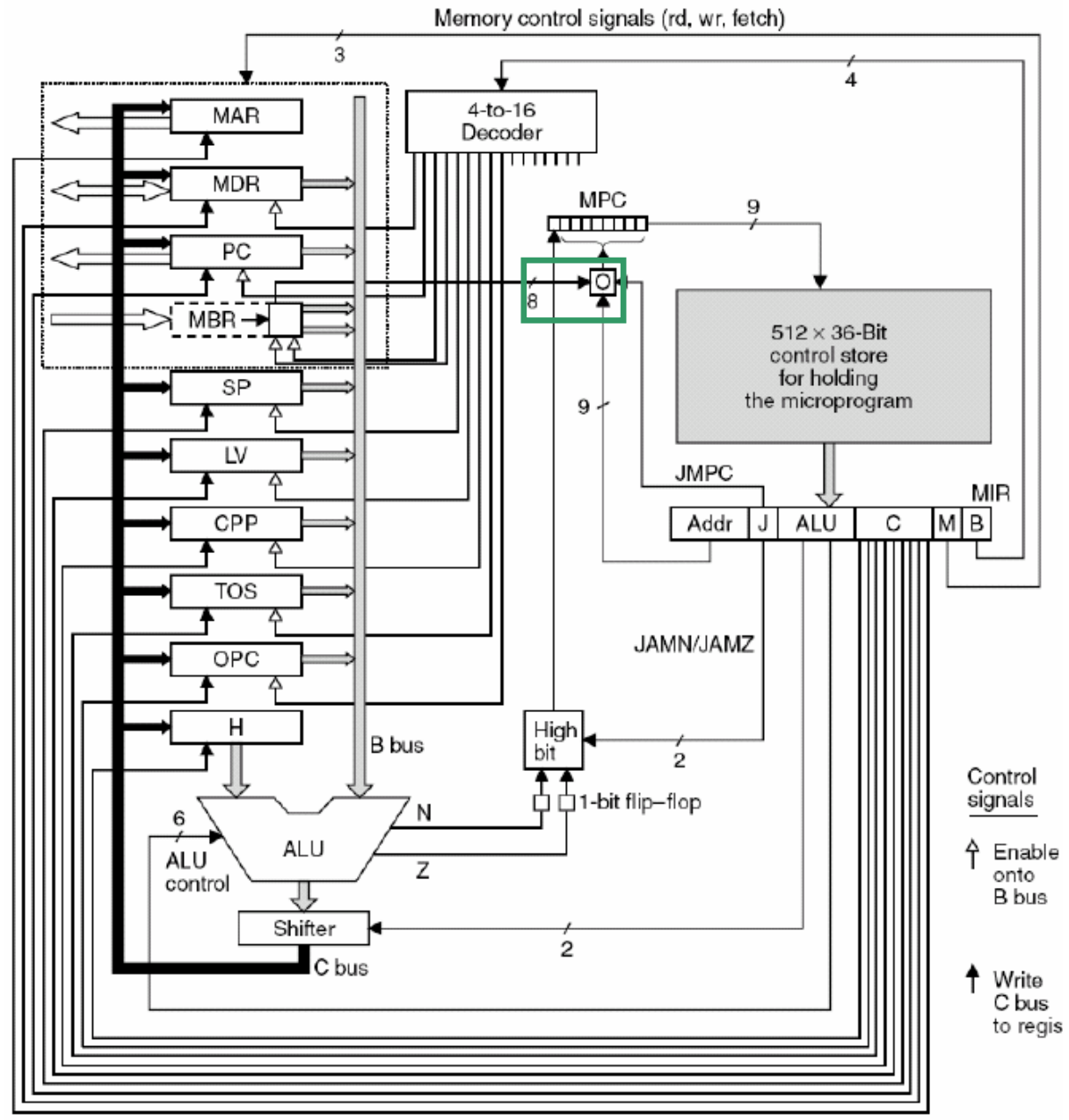
swap1 : MAR = SP - 1, rd (Og så utfør Next microinst addr)

MAR peker nå på element 2, og verdien 12 leses inn i MDR
Det vil ta en klokkesykel før verdien ligger i MDR





IJVM

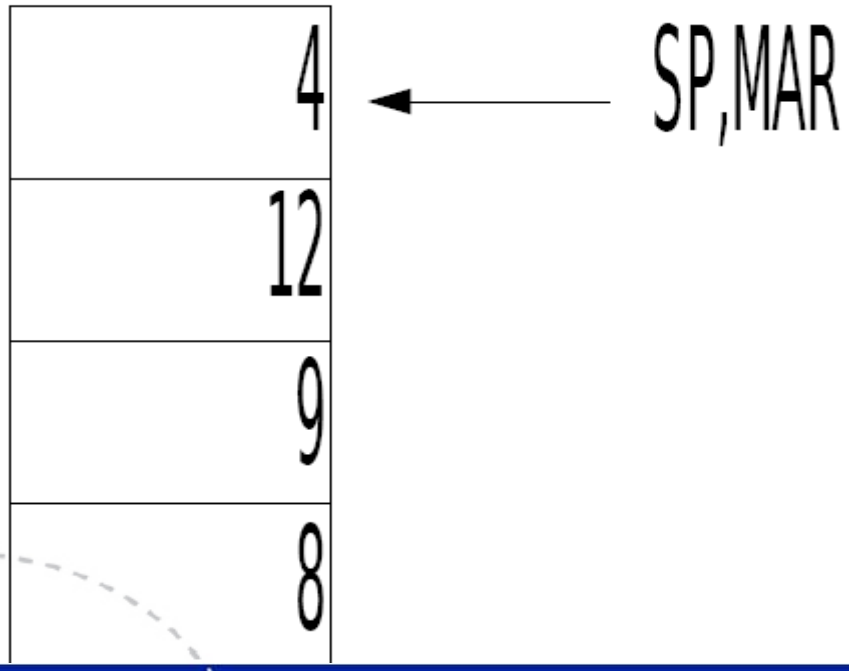

Memory control signals (rd, wr, fetch)
3
4
MAR
MDR
PC
MBR
SP
LV
CPP
TOS
OPC
H
4-to-16
Decoder
MPC
9
8
512 × 36-Bit
control store
for holding
the microprogram
9
JMPC
MIR
Addr
J
ALU
C
M
B
JAMN/JAMZ
B bus
High
bit
2
1-bit flip–flop
N
Z
6
ALU
control
ALU
Shifter
2
C bus
Control
signals
Enable
onto
B bus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

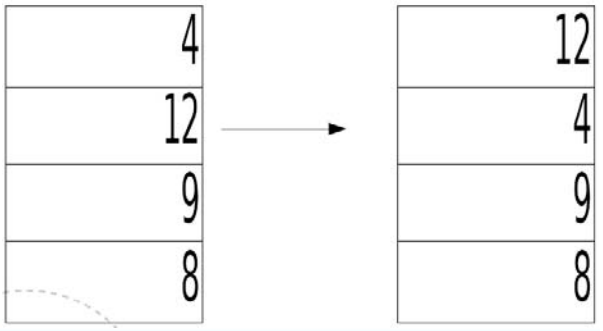
Microinstruksjon mem eksempel

Vil lagre MDR på toppen av stakken:

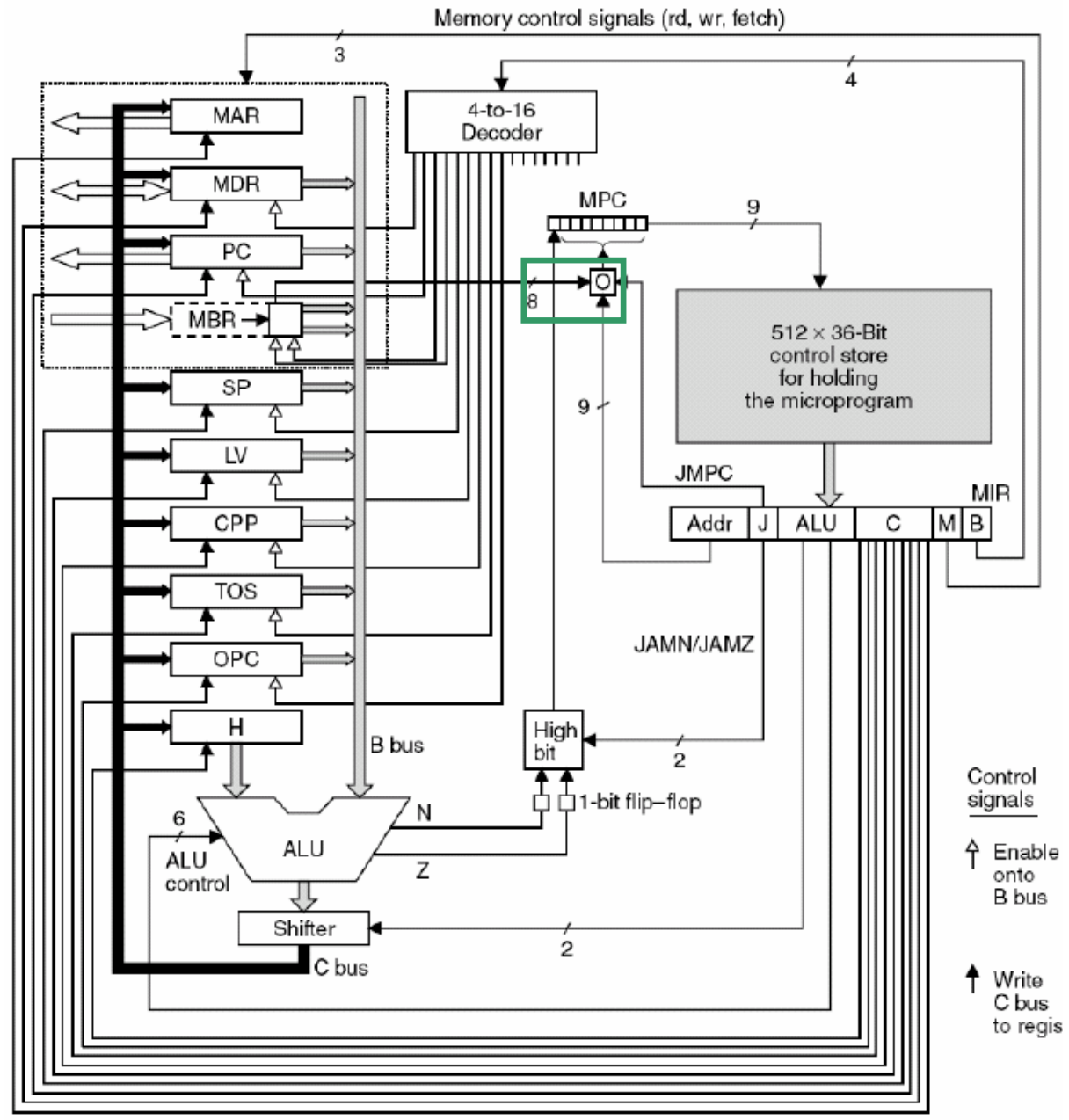
swap2 : MAR = SP

MAR peker nå på element 1
MEN - lesingen fra forrige mikroinstruksjon er enda ikke ferdig.





IJVM

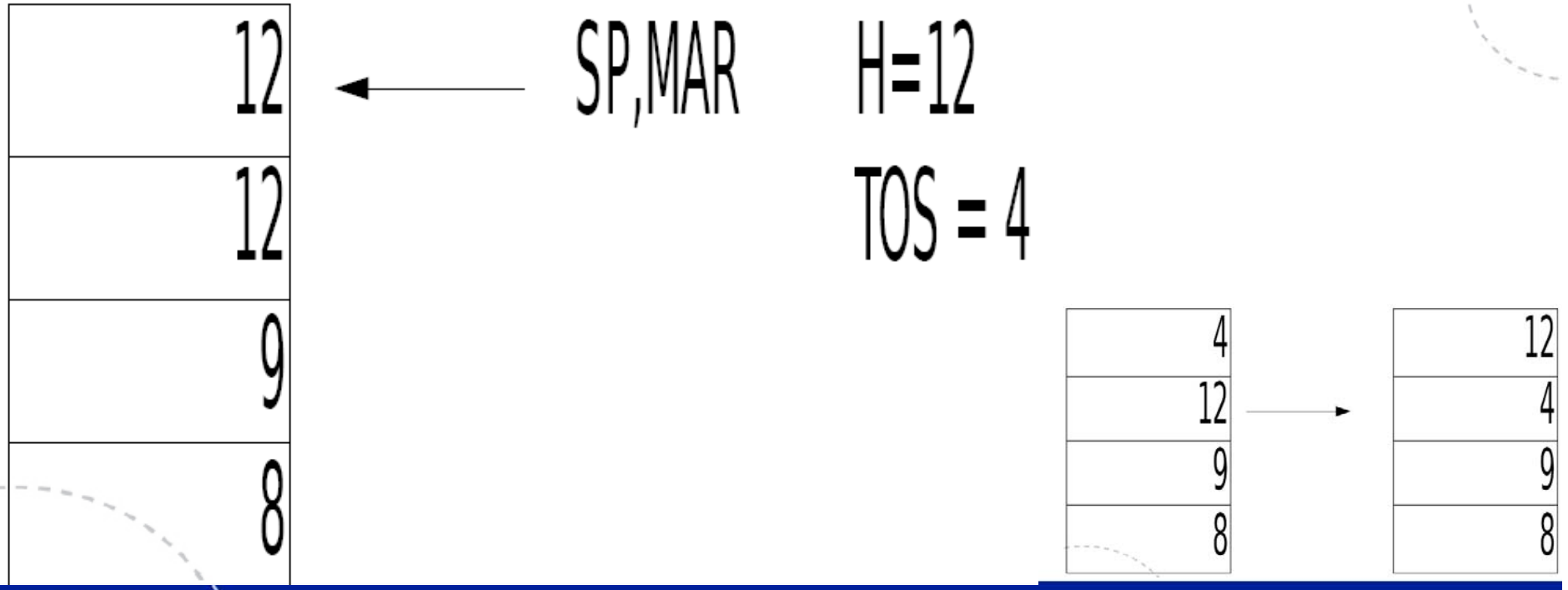

Memory control signals (rd, wr, fetch)
3
4
MAR
MDR
PC
MBR
SP
LV
CPP
TOS
OPC
H
4-to-16
Decoder
MPC
9
8
512 × 36-Bit
control store
for holding
the microprogram
9
JMPC
MIR
Addr
J
ALU
C
M
B
JAMN/JAMZ
High
bit
2
B bus
1-bit flip–flop
N
Z
6
ALU
control
ALU
Shifter
2
C bus
Control
signals
Enable
onto
B bus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

Microinstruksjon mem eksempel

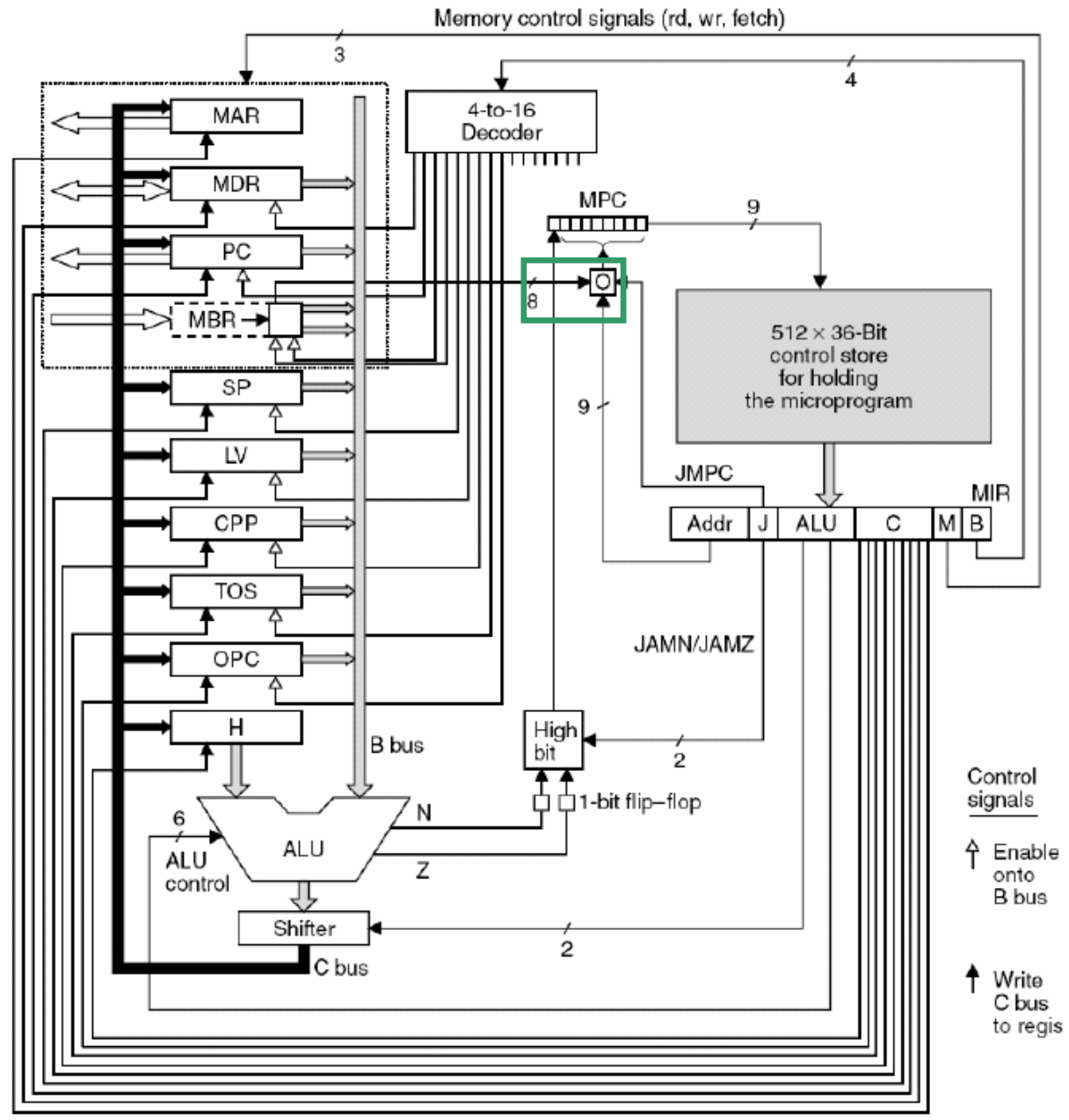
Mellomlagre 2.element av stakken:

swap3 : H = MDR, wr

Mellomlagre bunnen av stakken.
Skrive 2. element på toppen
Husk – TOS inneholder fortsatt det øverste elementet



IJVM

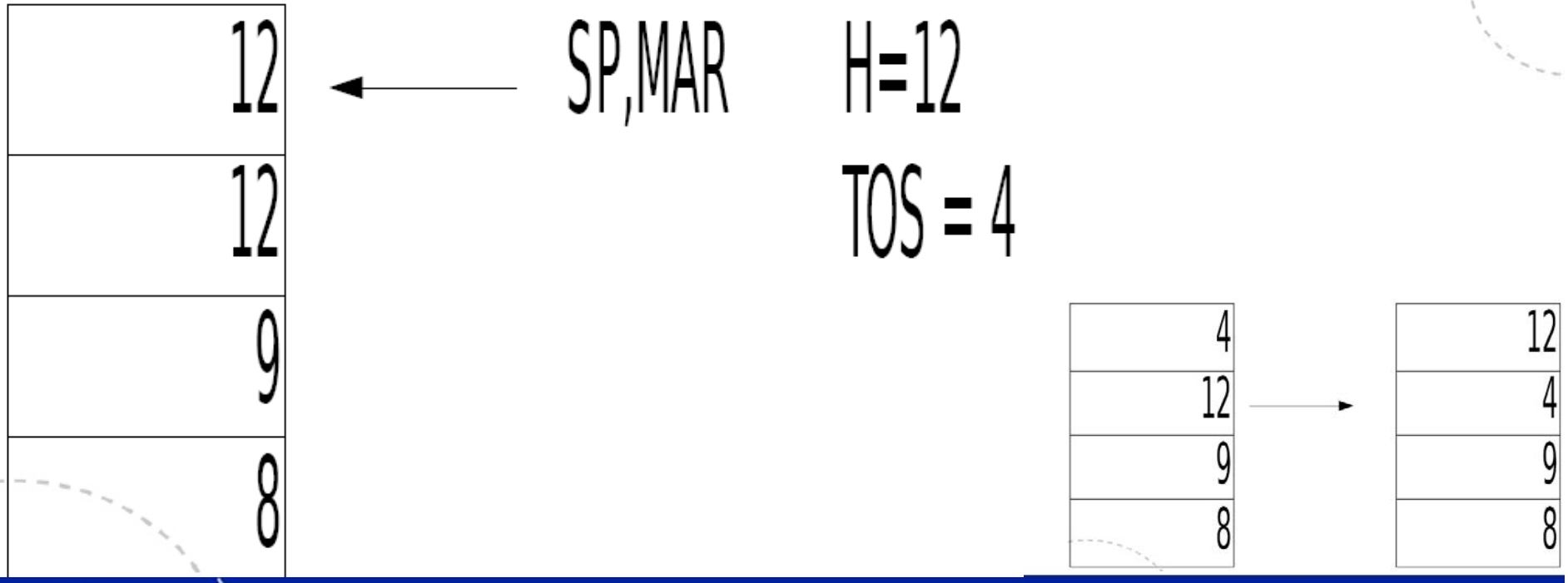

Memory control signals (rd, wr, fetch)
3
4
MAR
MDR
PC
MBR
SP
LV
CPP
TOS
OPC
H
4-to-16
Decoder
MPC
9
8
512 × 36-Bit
control store
for holding
the microprogram
9
JMPC
MIR
Addr
J
ALU
C
M
B
JAMN/JAMZ
B bus
High
bit
2
1-bit flip–flop
N
Z
6
ALU
control
ALU
Shifter
2
C bus
Control
signals
Enable
onto
B bus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

Microinstruksjon mem eksempel

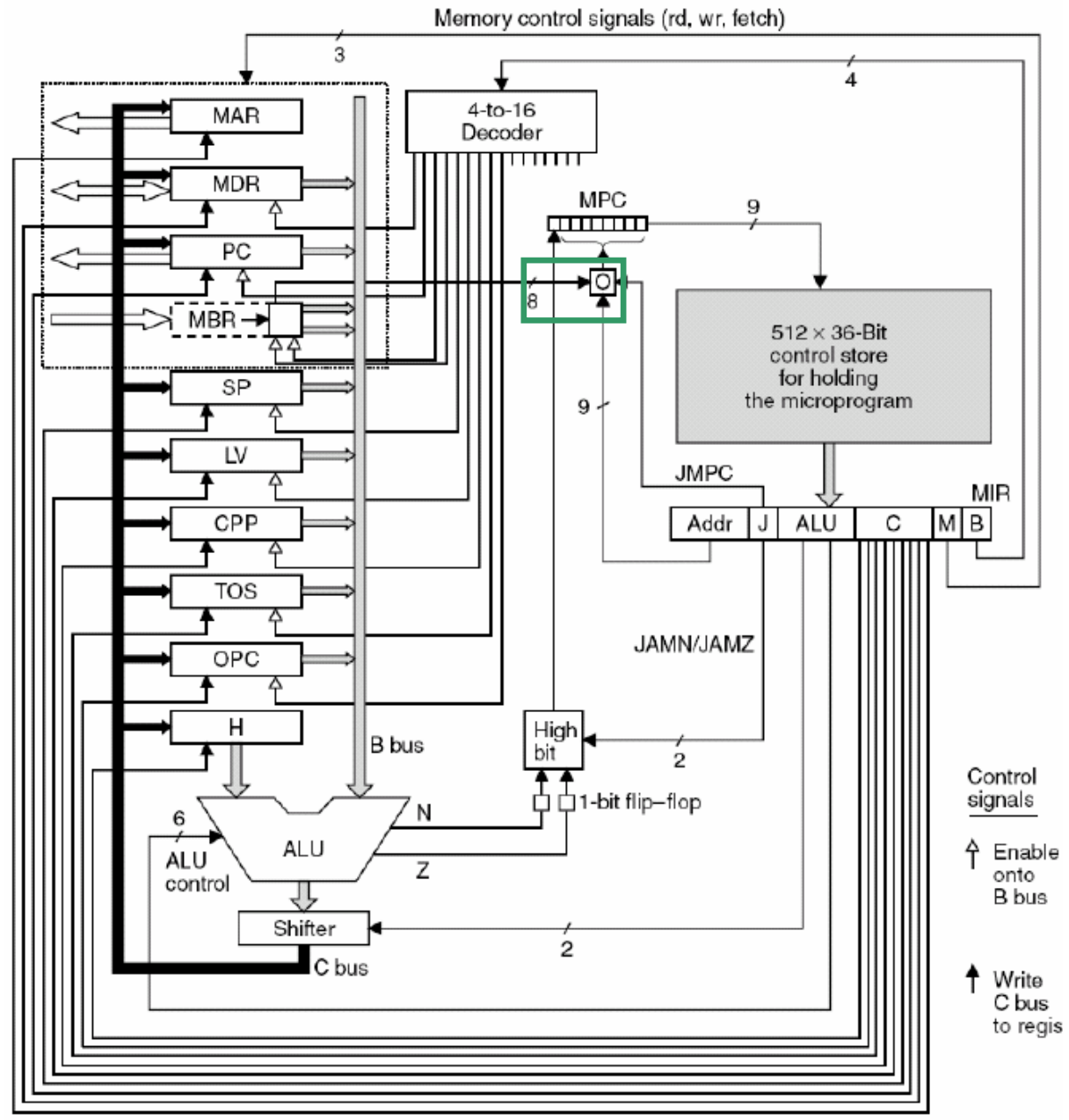
Forbered å skrive gamle TOS inn i plass 2

swap4 : MDR = TOS

Gamle TOS (4) legges i dataregisteret slik at vi kan skrive det ut



IJVM

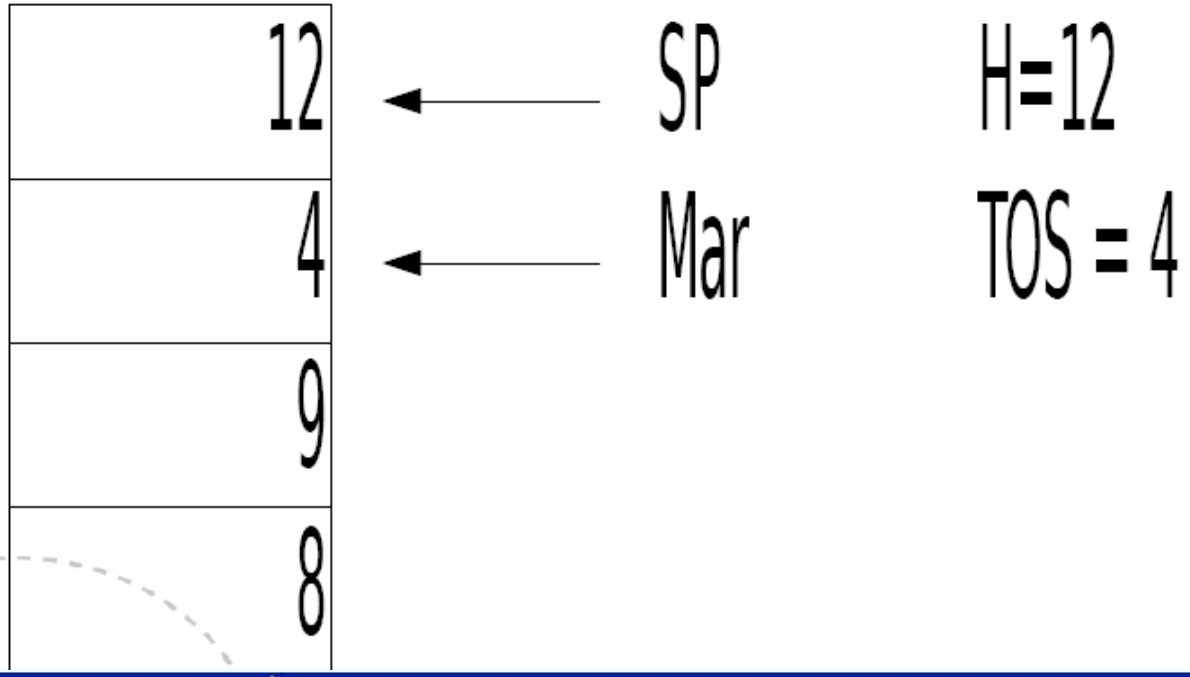

Memory control signals (rd, wr, fetch)
3
4
4-to-16 Decoder
MAR
MDR
PC
MBR
SP
LV
CPP
TOS
OPC
H
MPC
9
8
512 × 36-Bit control store for holding the microprogram
9
JMPC
MIR
Addr
J
ALU
C
M
B
JAMN/JAMZ
High bit
2
B bus
N
Z
1-bit flip–flop
6
ALU control
ALU
Shifter
2
C bus
Control signals
Enable onto B bus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

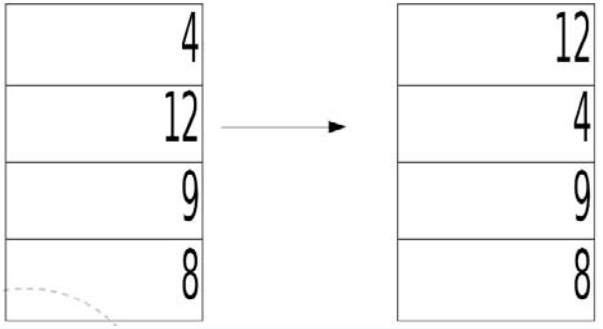
Microinstruksjon mem eksempel

Skriv gamle TOS til posisjon 2

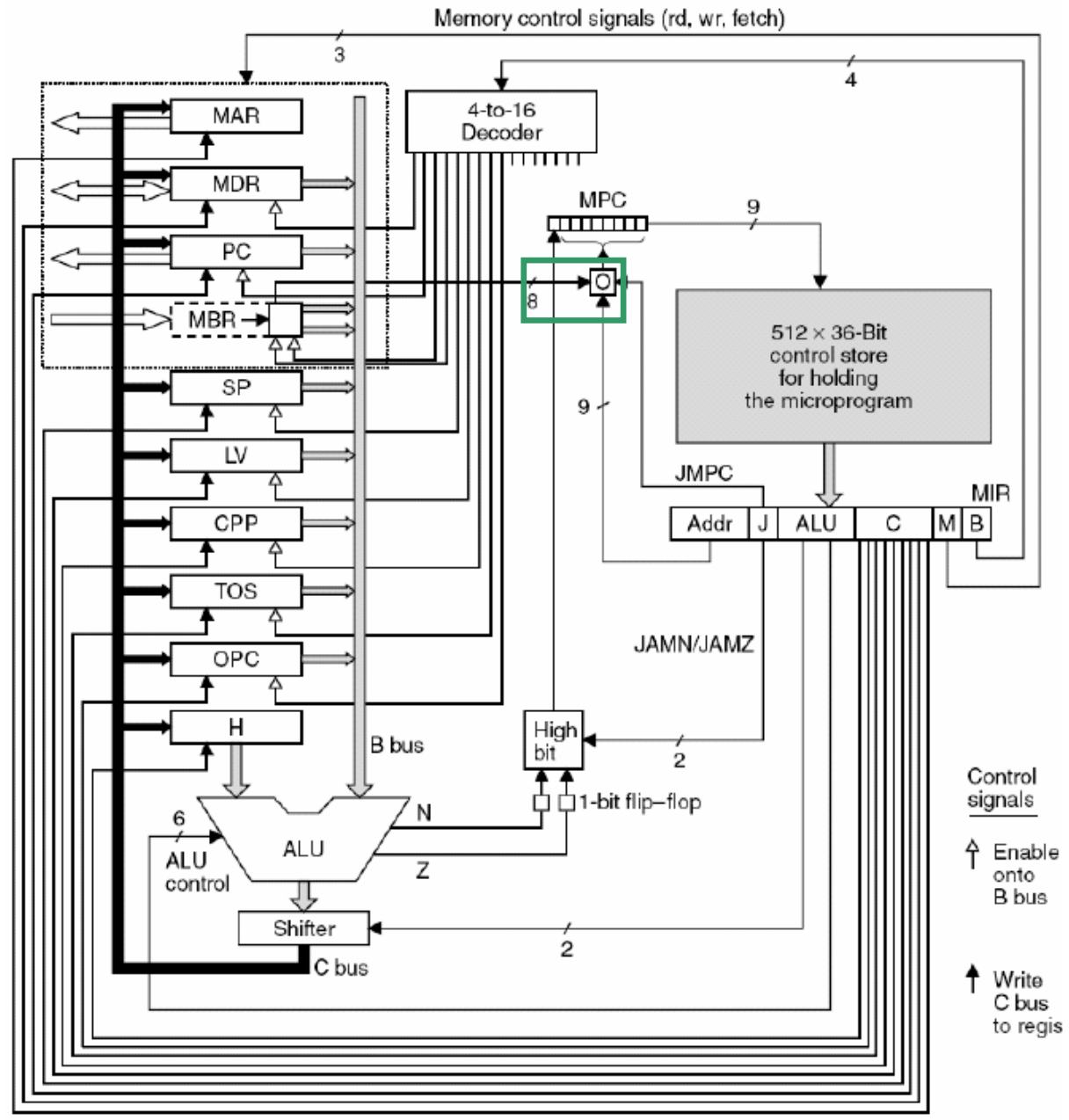
swap5 : MAR = SP – 1, wr

MAR peker på element 2 i stakken
Skriv MDR (gamle TOS) på den posisjonen





IJVM

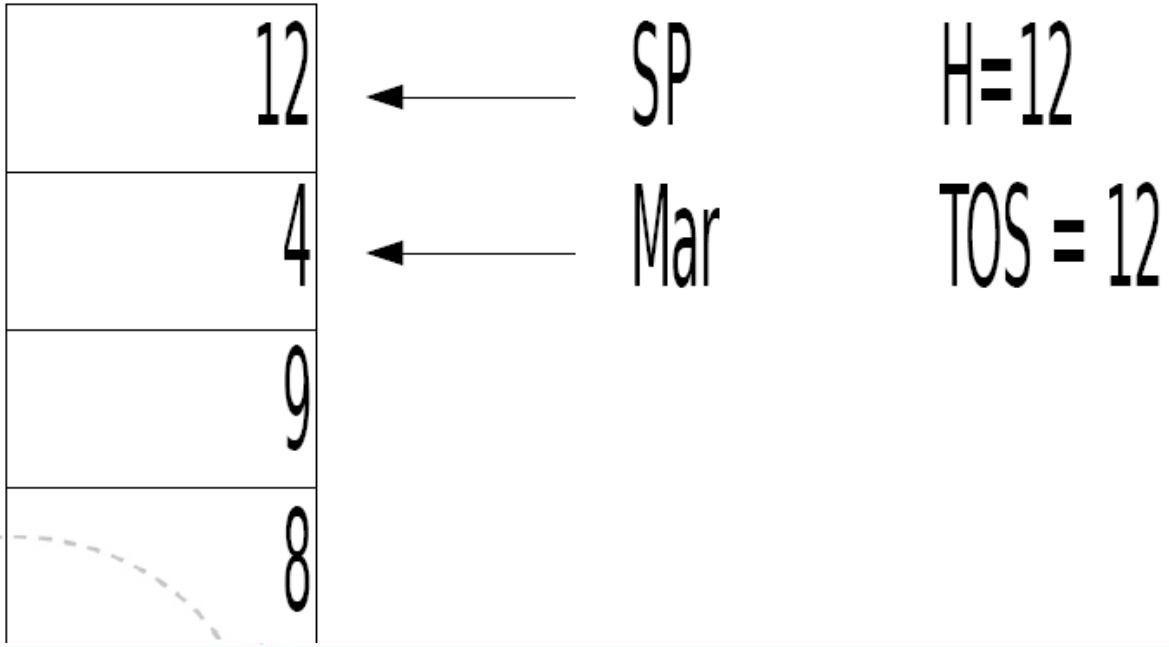

Memory control signals (rd, wr, fetch)
3
4
MAR
MDR
PC
MBR
SP
LV
CPP
TOS
OPC
H
4-to-16
Decoder
MPC
9
8
512 × 36-Bit
control store
for holding
the microprogram
9
JMPC
MIR
Addr
J
ALU
C
M
B
JAMN/JAMZ
B bus
High
bit
2
1-bit flip–flop
N
Z
6
ALU
control
ALU
Shifter
2
C bus
Control
signals
Enable
onto
B bus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

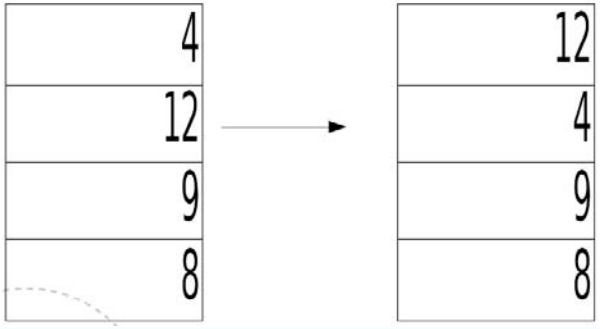
Microinstruksjon mem eksempel

Oppdater TOS

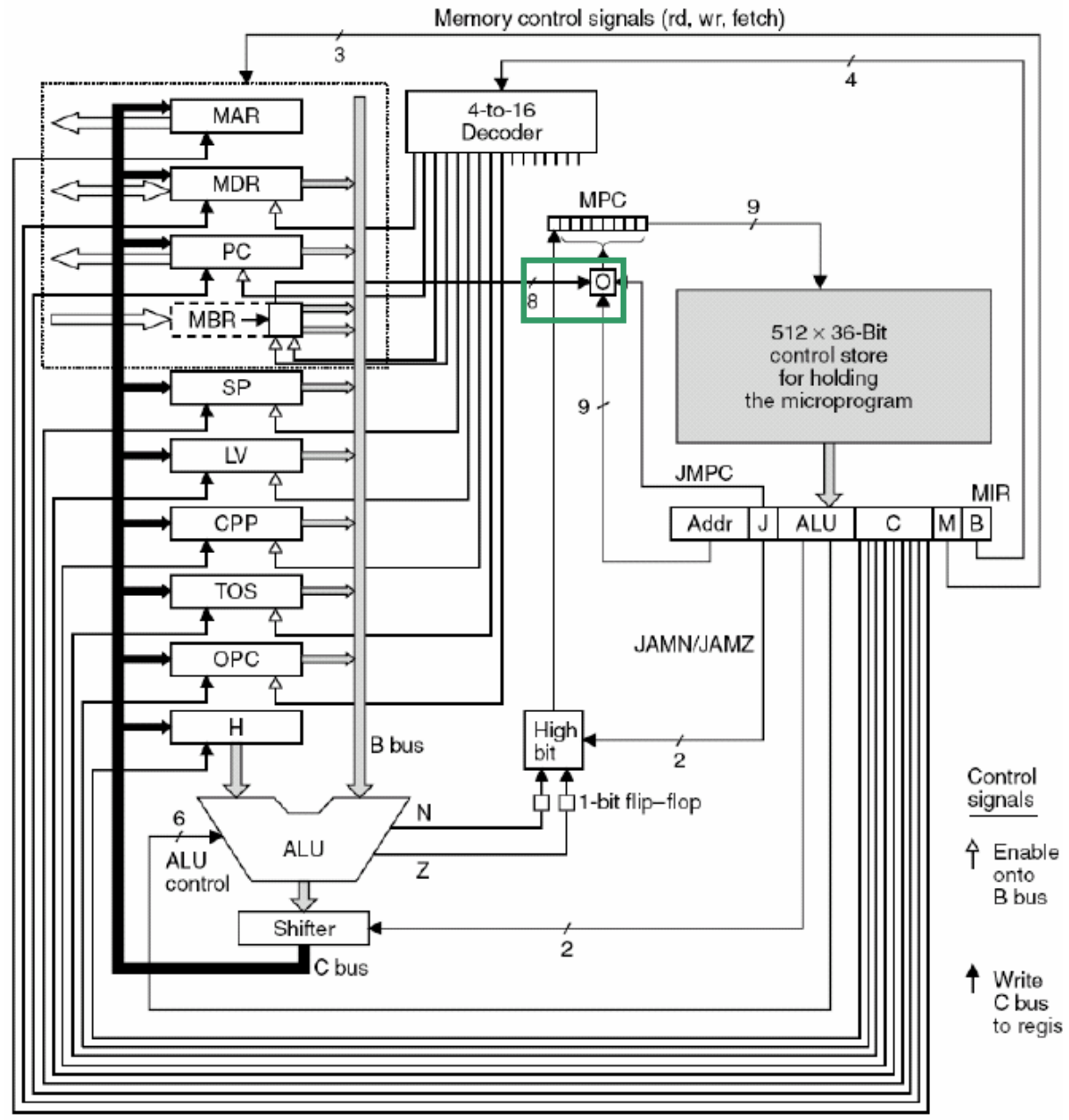
swap6 : TOS = H; Goto Main 1

TOS skal alltid inneholde elementet på toppen av stakken når instruksjonen er fullført.
Gå til Main1 for å starte utførselen av neste instruksjon



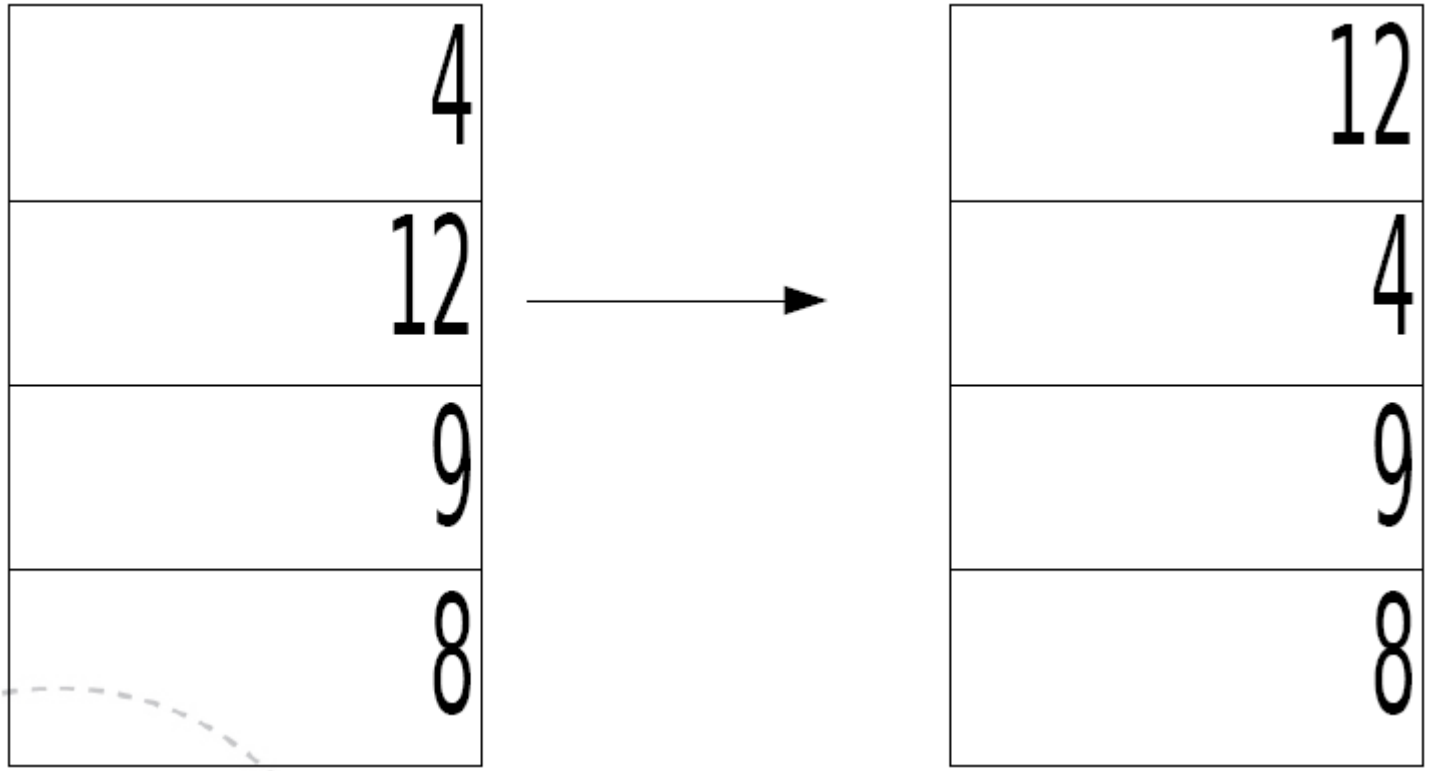


IJVM


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 regis

Swap: opCod 0x5F

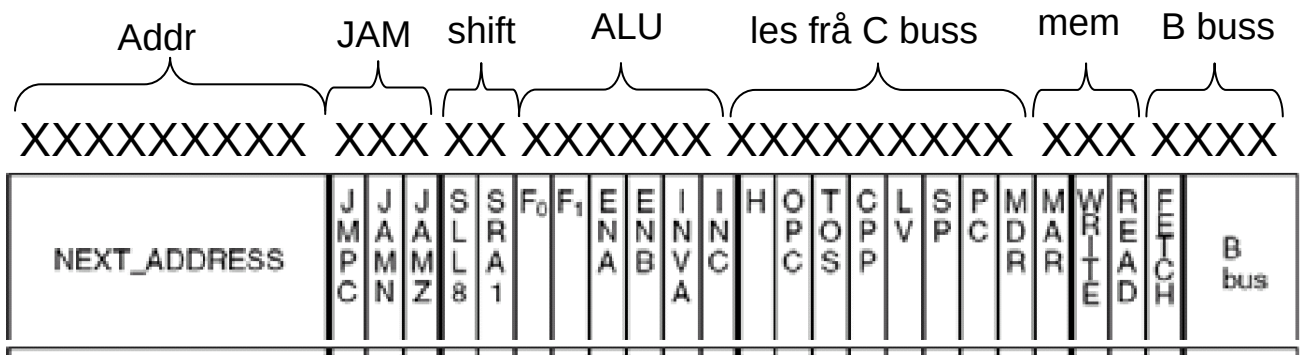
- swap1 : MAR = SP - 1, rd
- swap2 : MAR = SP
- swap3 : H = MDR, wr
- swap4 : MDR = TOS
- swap5 : MAR = SP – 1, wr
- swap6 : TOS = H; Goto Main 1

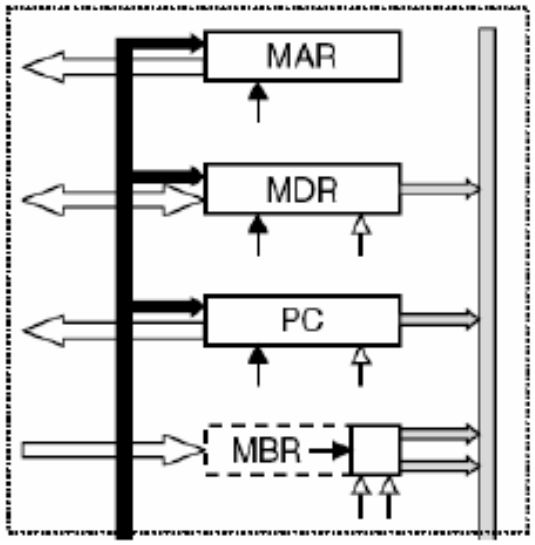


NTNU
Innovation and Creativity

Swap: opCod 0x5F

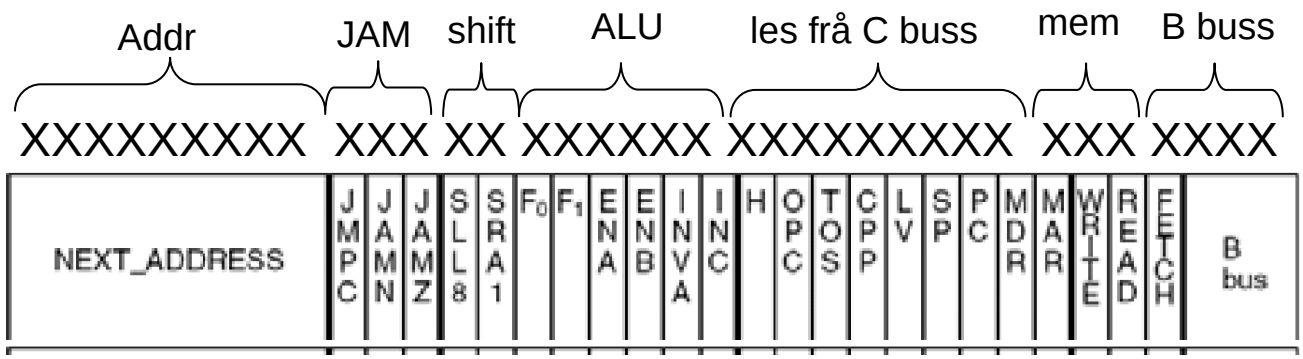
- MPC: 0x5F
- swap1 (0x5F): MAR = SP - 1, rd
  - (Addr: 0x60, JAM: 0, Shif: 0, ALU: 0x36 (b-1), cBuss: 0x01 (SP), **mem: ”010” (rd)**, bBuss: 0x4 (SP))
- swap2 (0x60): MAR = SP
- swap3 (0x61): H = MDR, wr
- swap4 (0x62): MDR = TOS
- swap5 (0x63): MAR = SP – 1, wr
- swap6 (0x00): TOS = H; Goto Main 1 (Hent neste instruksjon)

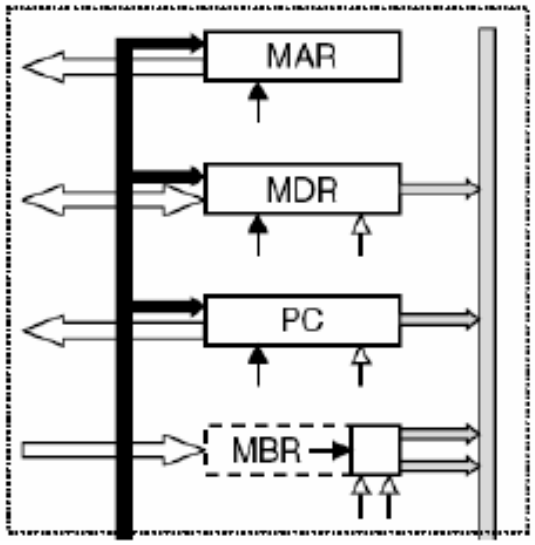




Hent neste instruksjon

- swap6 (0x01): TOS = H; Goto Main 1 (Hent neste instruksjon)
  - 0x00: PC = PC +1, fetch
    - Addr: 0x01,
    - JAM : 0,
    - Shift: 0,
    - ALU: 0x35 (b + 1),
    - cBuss: 0x04 (PC),
    - mem: ”001”,
    - bBuss: 0x04 (SP)
  - Goto Main 1 Addr: 0x00 (ubetinga hopp i microkoden)



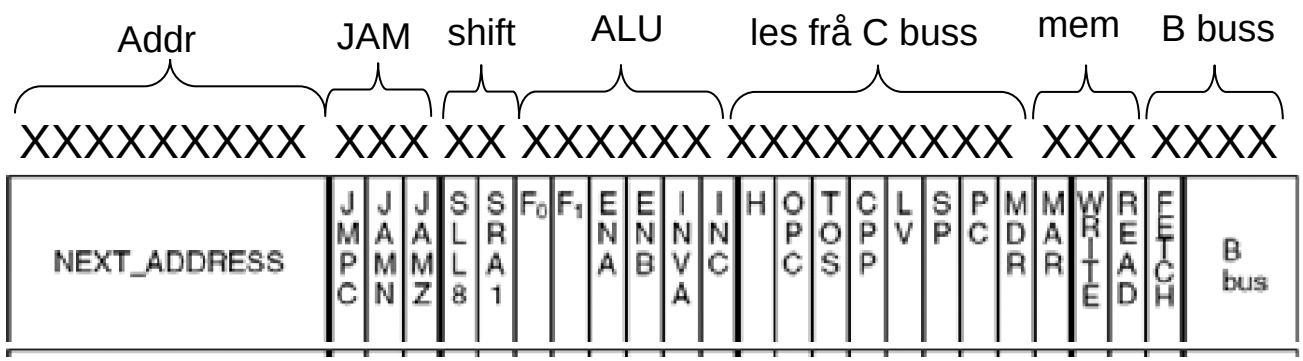


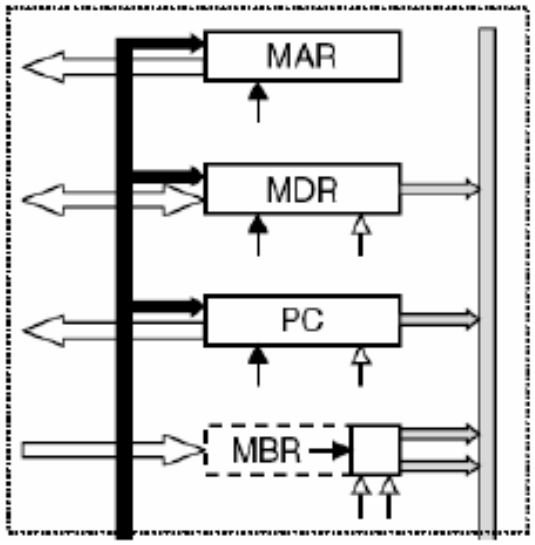
Hent neste instruksjon

- swap6 (0x00): TOS = H; Goto Main 1 (Hent neste instruksjon)
  - **Main1**: 0x00: PC = PC +1, fetch, Addr: 0X01
  - 0x01: NOP (ingen lesing skriving av register), Addr: 0x01

:

  - swap6 (0x00): TOS = H; **Goto Main 1** (Alle instruksjonar avslutast slik)

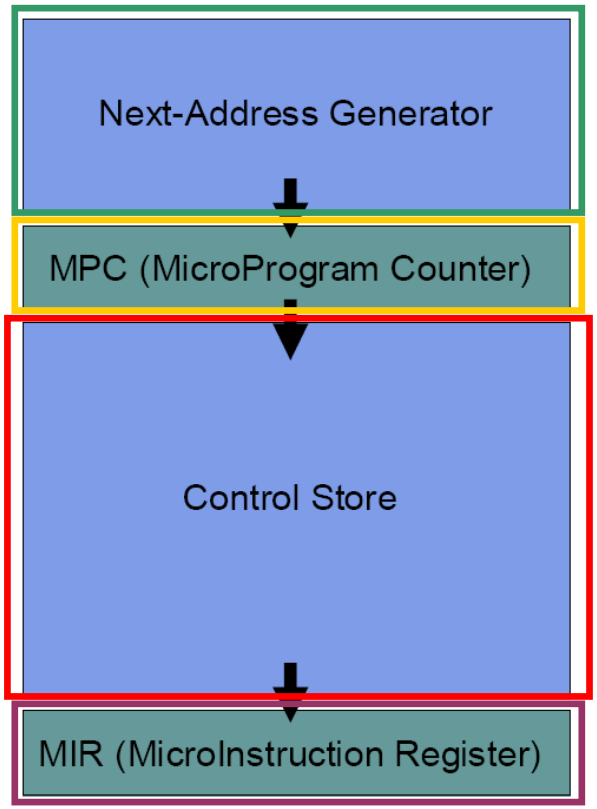




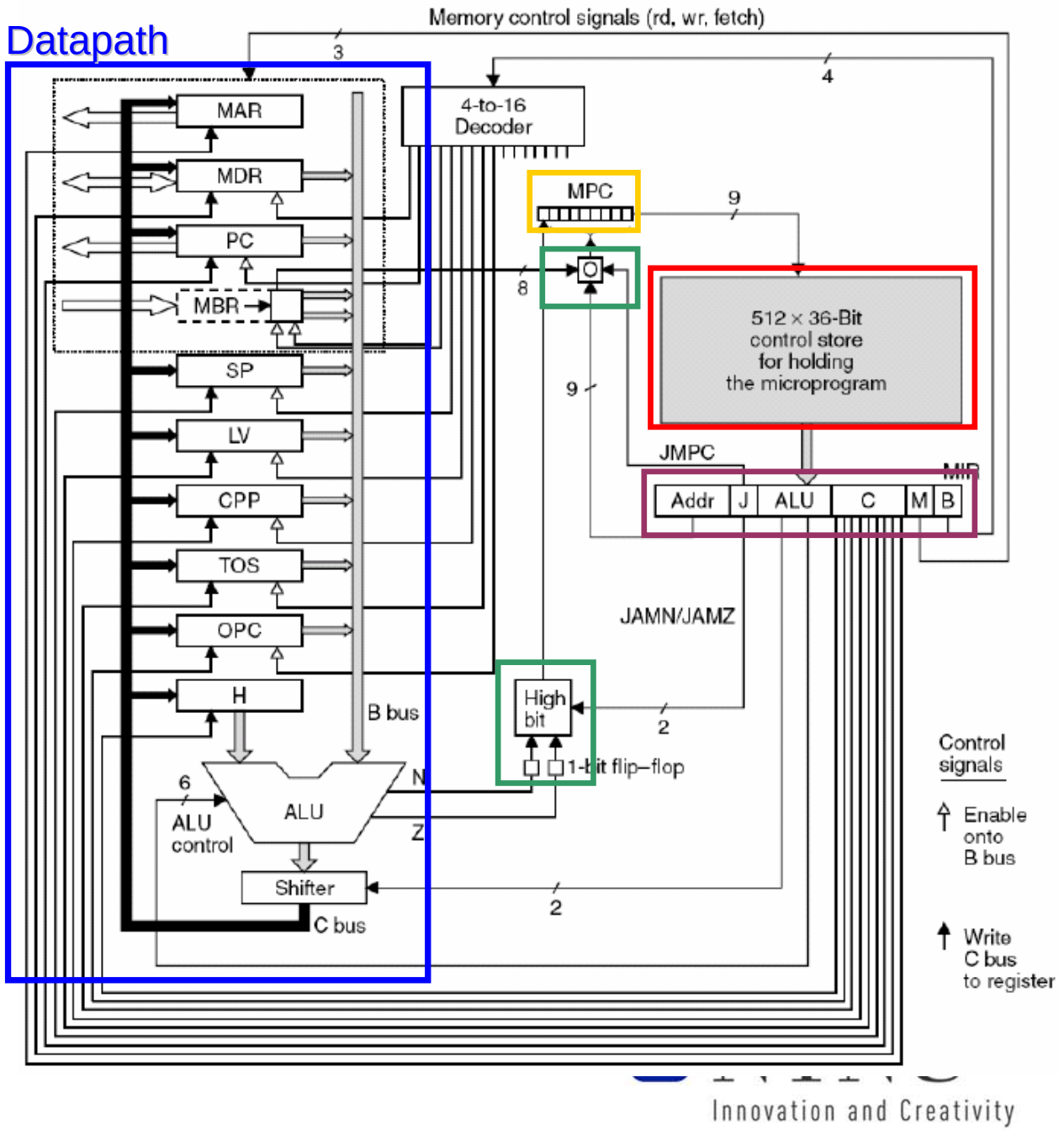
Register bruk

- CPP, LV, SP holder pekerverdier
- PC adresse til neste byte med programkode
- MBR siste byte lest fra programkode
- MAR/MDR adresse til data / data
- TOS skal alltid inneholde ord på topp av stakk
- OPC register til fritt bruk

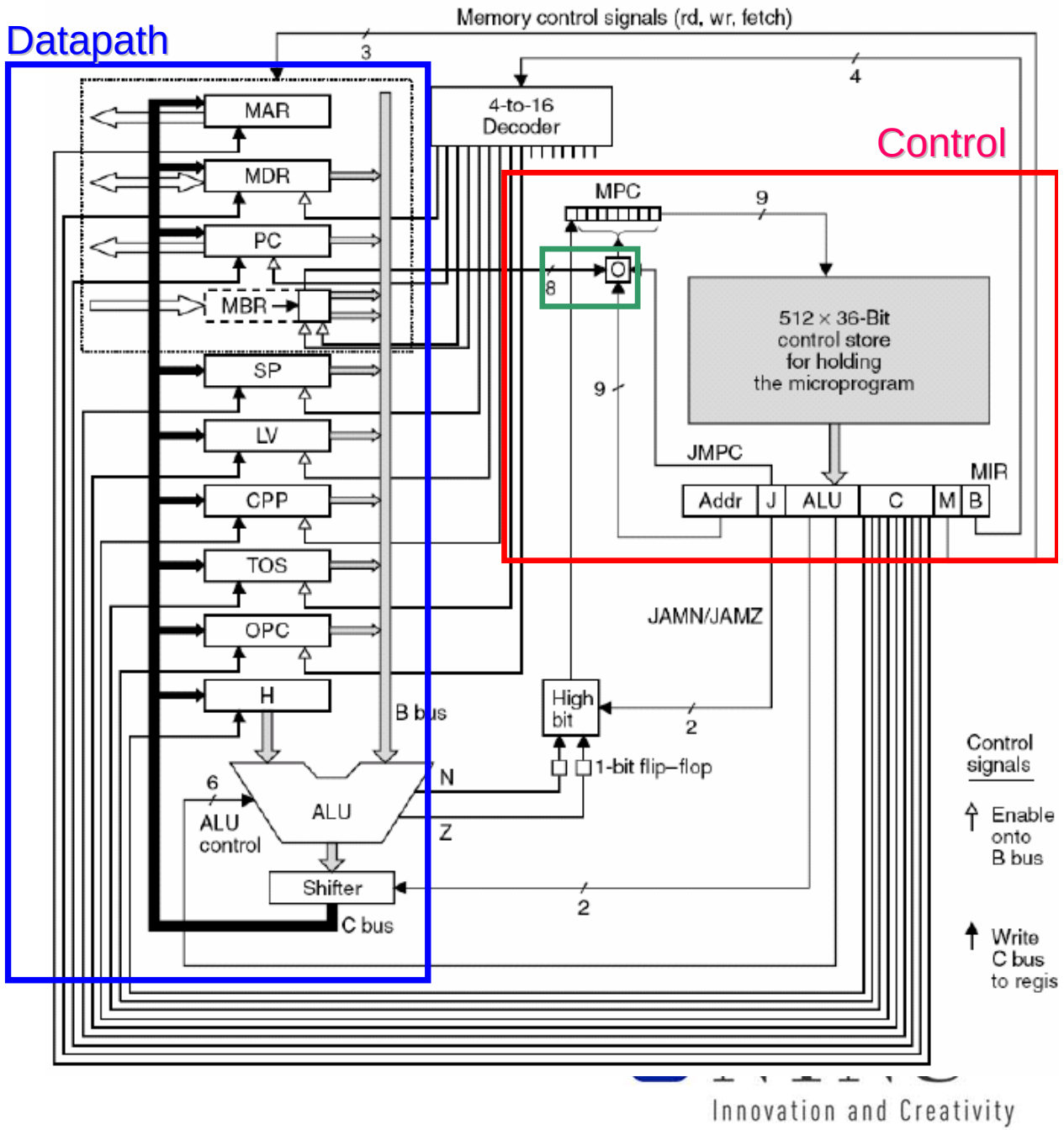
IJVM



Datapath



IJVM


Datapath
Control
Memory control signals (rd, wr, fetch)
3
4
4-to-16
Decoder
MAR
MDR
PC
MBR
SP
LV
CPP
TOS
OPC
H
B bus
MPC
9
8
512 × 36-Bit
control store
for holding
the microprogram
9
JMPC
MIR
Addr
J
ALU
C
M
B
JAMN/JAMZ
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 regis

IJVM

- Implementasjon
- Kva kan gjerast for å auke ytinga