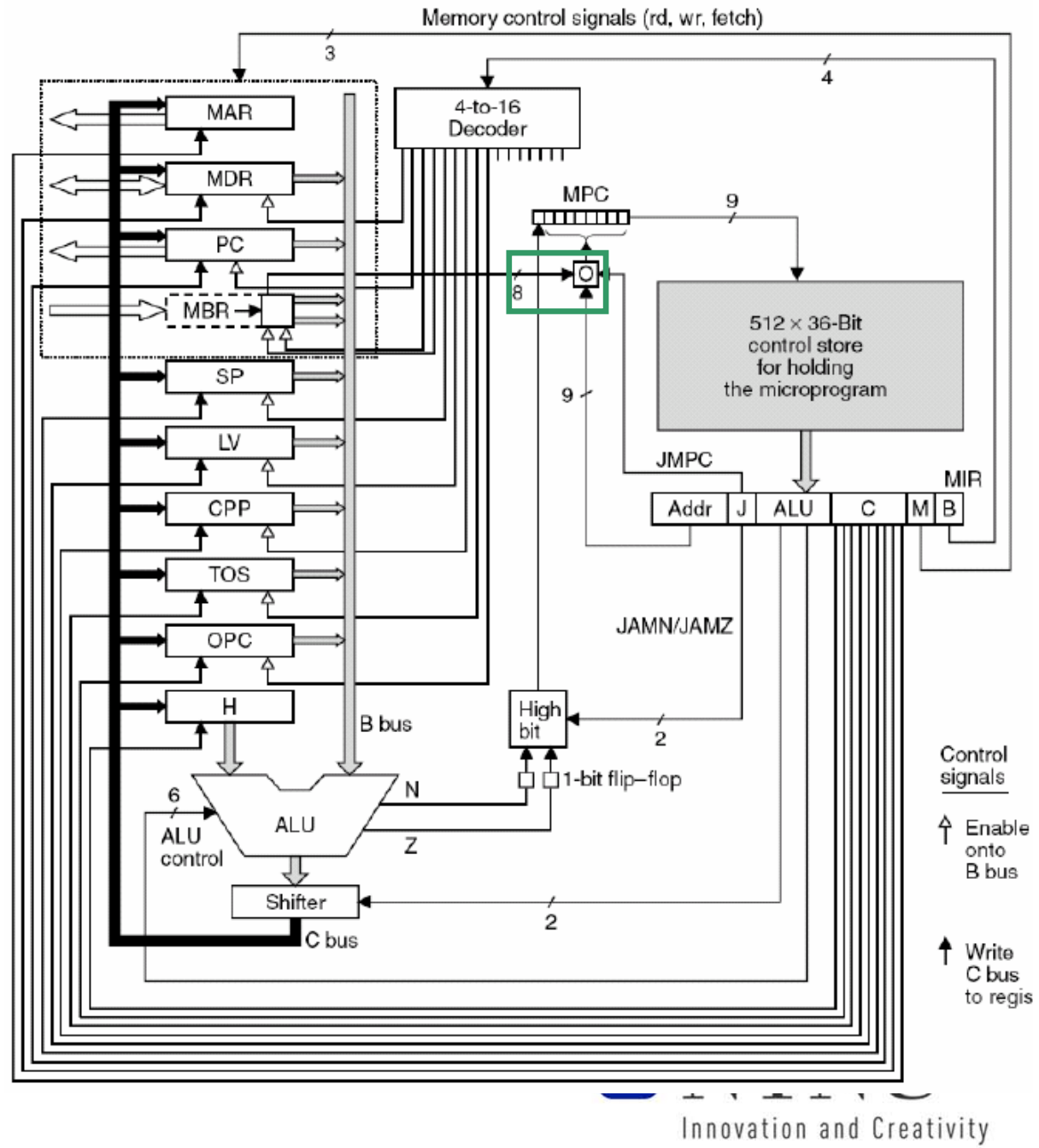
Fortsetelse Microarchitecture level

IJVM

- Implementasjon
- Kva kan gjerast for å auke ytinga

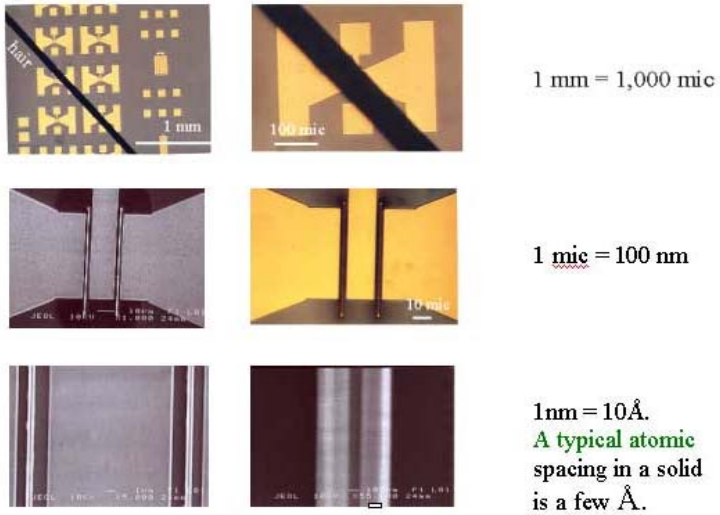


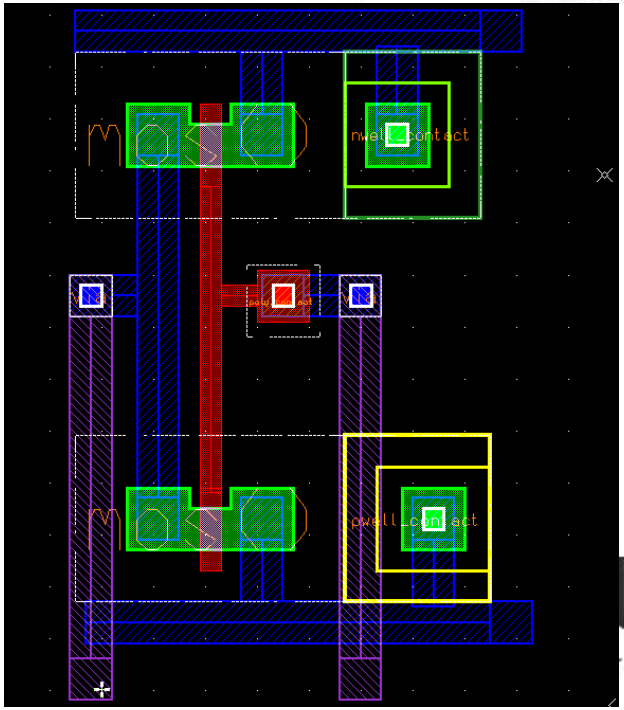
IJVM

- Implementasjon
  - Detaljar for å utføre instruksjonssettet
  - Ein gitt implementasjon har ein gitt yting
  - Endre ytinga
    - Teknologi (prosess)
    - Transistor implementasjon
    - Digital design av komponentar
    - Auke mengda logikk
    - Endring Av arkitektur
- Kva kan gjerast for å auke ytinga

IJVM

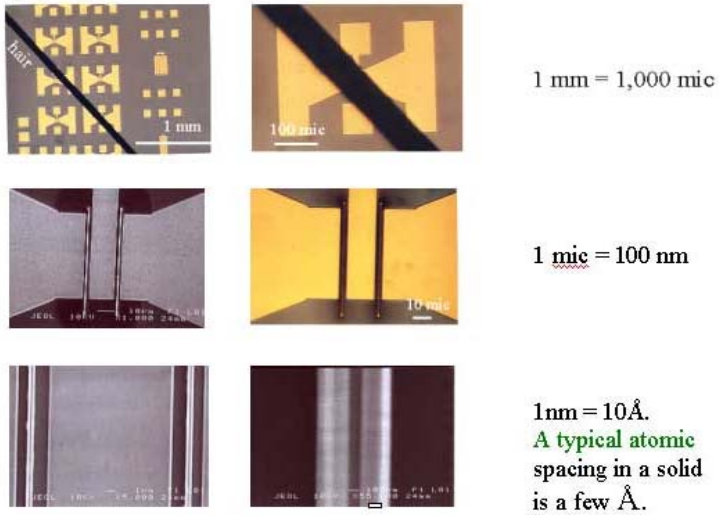
- Implementasjon
  - Detaljar for å utføre instruksjonssettet
  - Ein gitt implementasjon har ein gitt yting
  - Endre ytinga
    - Teknologi (prosess)
    - Transistor implementasjon
    - Digital design av komponentar
    - Auke mengda logikk
    - Endring Av arkitektur
- Kva kan gjerast for å auke ytinga

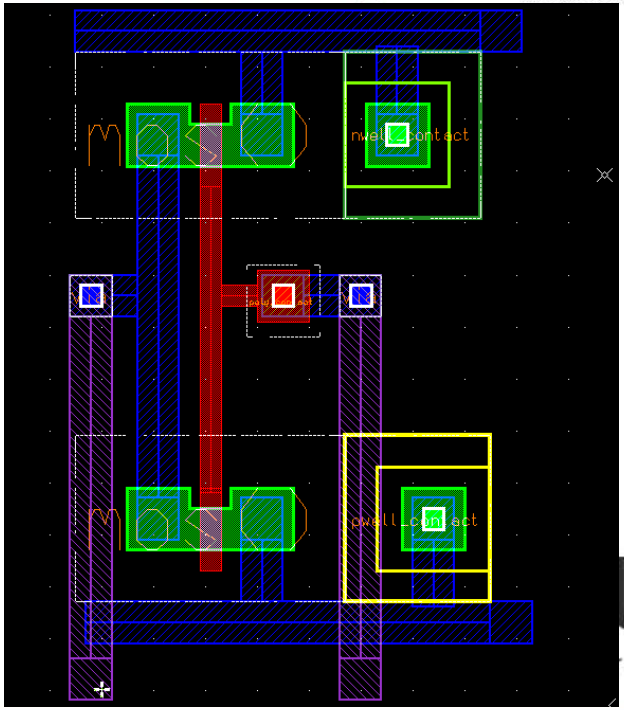




IJVM

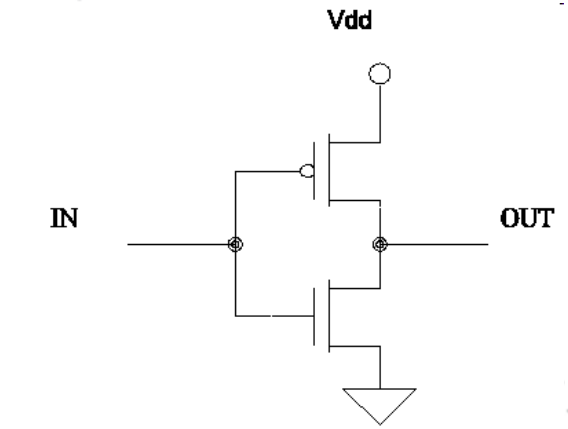
- Implementasjon
  - Detaljar for å utføre instruksjonssettet
  - Ein gitt implementasjon har ein gitt yting
  - Endre ytinga
    - Teknologi (prosess)
    - Transistor implementasjon
    - Digital design av komponentar
    - Auke mengda logikk
    - Endring Av arkitektur
- Kva kan gjerast for å auke ytinga

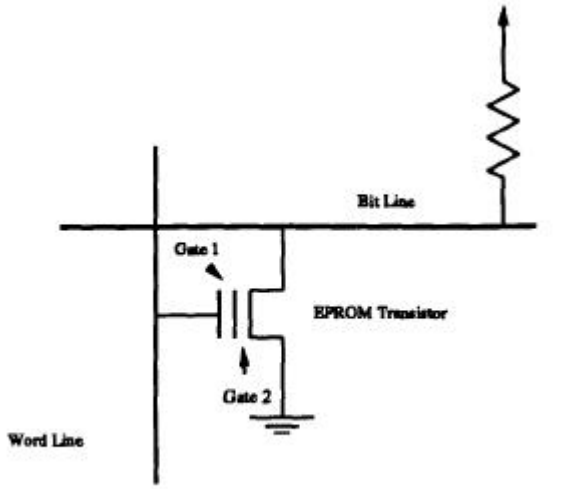




IJVM

- Implementasjon
  - Detaljar for å utføre instruksjonssettet
  - Ein gitt implementasjon har ein gitt yting
  - Endre ytinga
    - Teknologi (prosess)
    - Transistor implementasjon
    - Digital design av komponentar
    - Auke mengda logikk
    - Endring Av arkitektur
- Kva kan gjerast for å auke ytinga



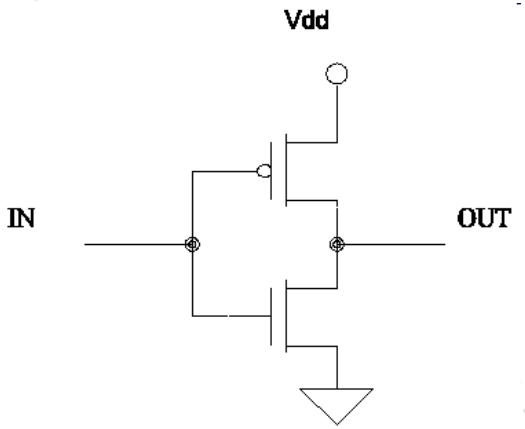


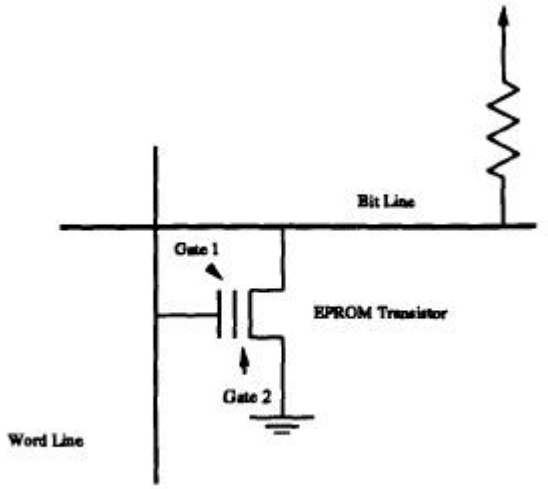
Floating gate programming technology.

NTNU
Innovation and Creativity

IJVM

- Implementasjon
  - Detaljar for å utføre instruksjonssettet
  - Ein gitt implementasjon har ein gitt yting
  - Endre ytinga
    - Teknologi (prosess)
    - Transistor implementasjon
    - Digital design av komponentar
    - Auke mengda logikk
    - Endring Av arkitektur
- Kva kan gjerast for å auke ytinga



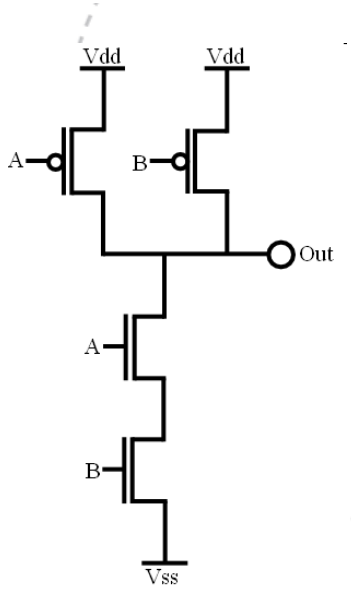


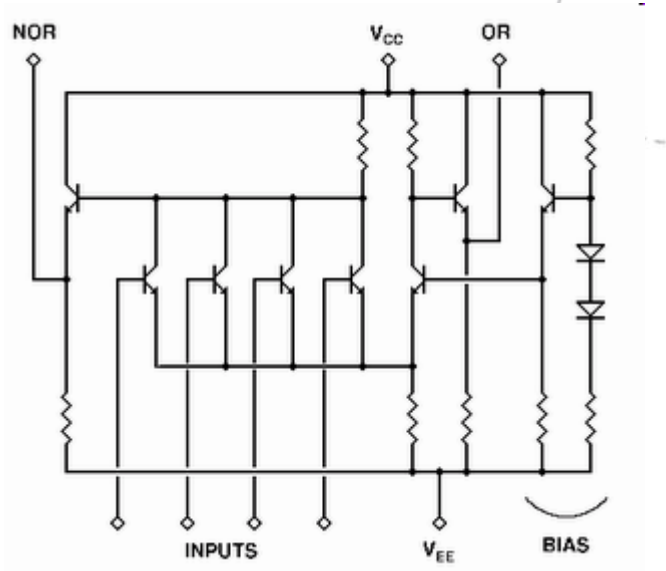
Floating gate programming technology.

NTNU
Innovation and Creativity

IJVM

- Implementasjon
  - Detaljar for å utføre instruksjonssettet
  - Ein gitt implementasjon har ein gitt yting
  - Endre ytinga
    - Teknologi (prosess)
    - Transistor implementasjon
    - Digital design av komponentar
    - Auke mengda logikk
    - Endring Av arkitektur
- Kva kan gjerast for å auke ytinga

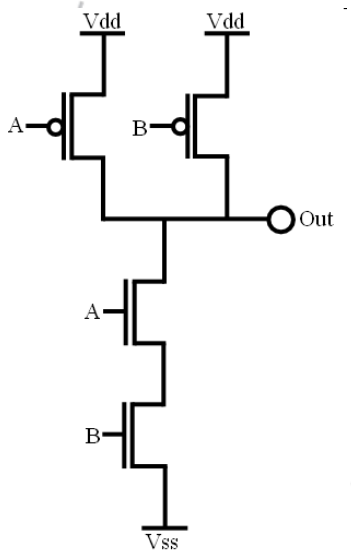


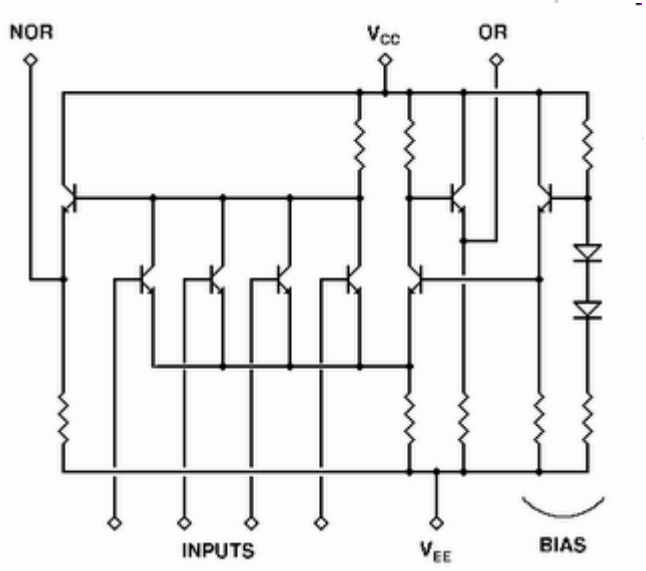


NTNU
Innovation and Creativity

IJVM

- Implementasjon
  - Detaljar for å utføre instruksjonssettet
  - Ein gitt implementasjon har ein gitt yting
  - Endre ytinga
    - Teknologi (prosess)
    - Transistor implementasjon
    - Digital design av komponentar
    - Auke mengda logikk
    - Endring Av arkitektur
- Kva kan gjerast for å auke ytinga

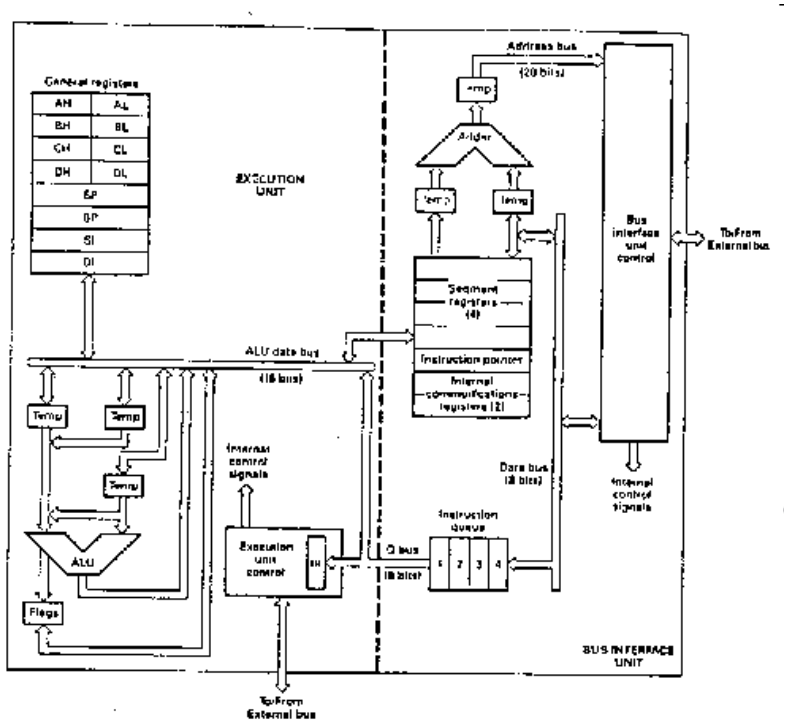


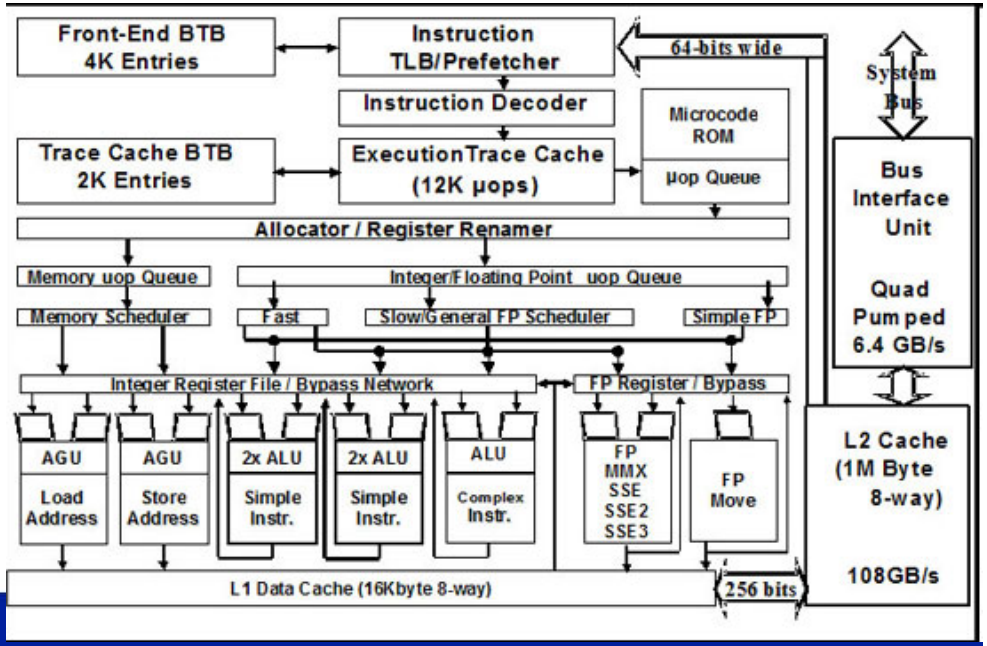


NTNU
Innovation and Creativity

IJVM

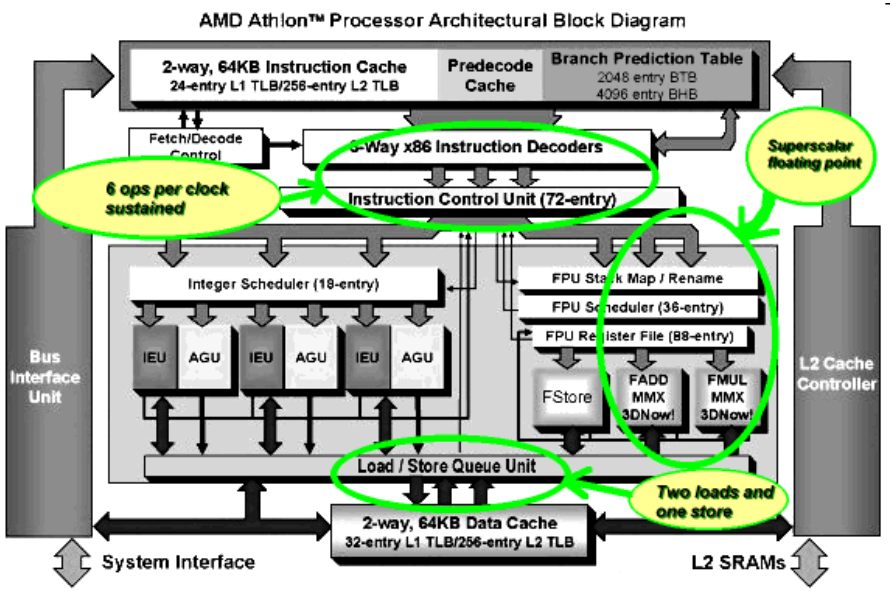
- Implementasjon
  - Detaljar for å utføre instruksjonssettet
  - Ein gitt implementasjon har ein gitt yting
  - Endre ytinga
    - Teknologi (prosess)
    - Transistor implementasjon
    - Digital design av komponentar
    - Auke mengda logikk
    - Endring Av arkitektur
- Kva kan gjerast for å auke ytinga



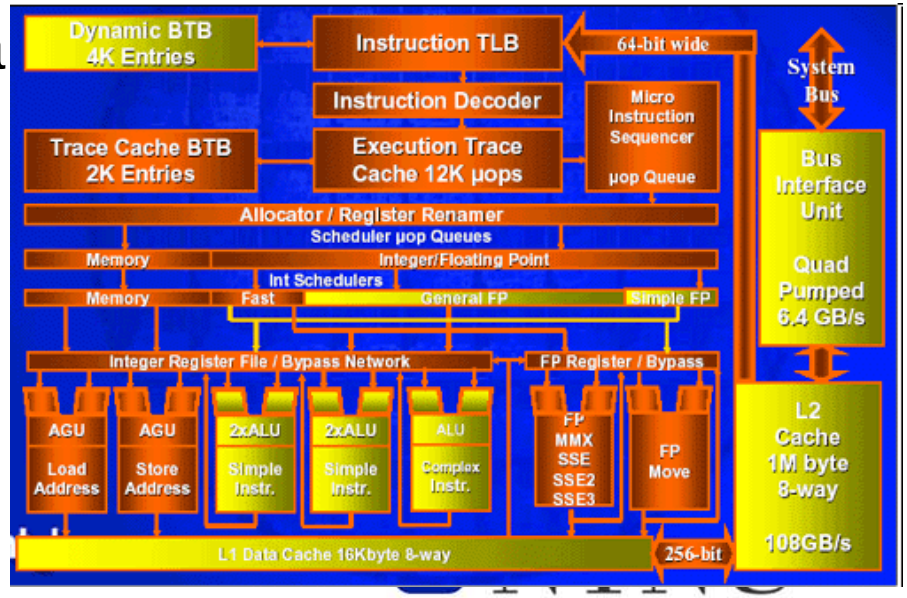


IJVM

- Implementasjon
  - Detaljar for å utføre instruksjonssettet
  - Ein gitt implementasjon har ein gitt yting
  - Endre ytinga
    - Teknologi (prosess)
    - Transistor implementasjon
    - Digital design av komponentar
    - Auke mengda logikk
    - Endring Av arkitektur
- Kva kan gjerast for å auke ytinga



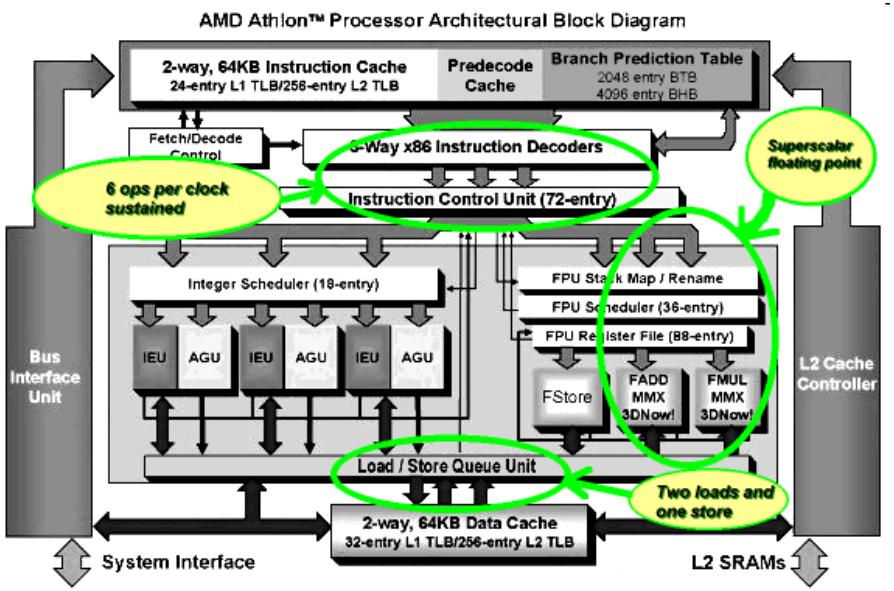
The AMD Athlon



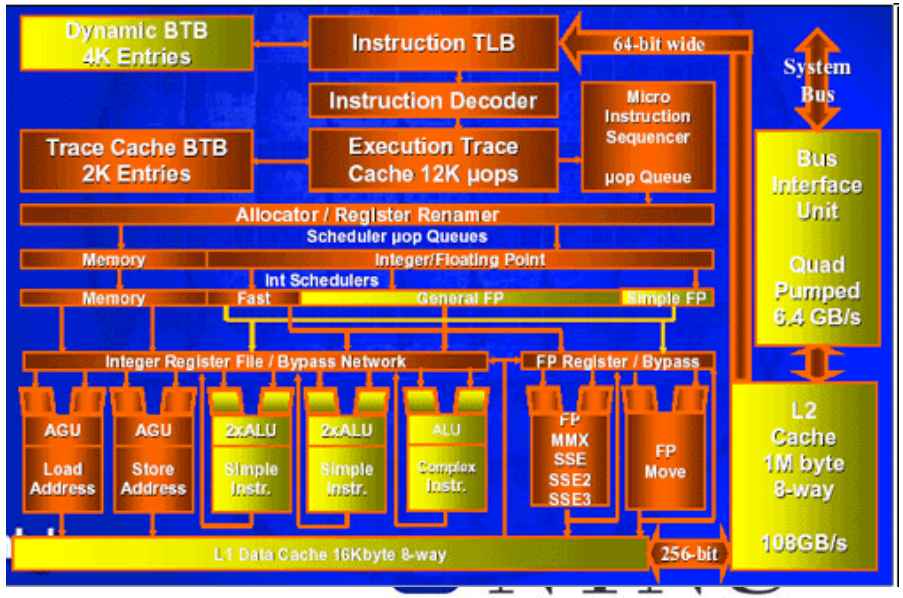
Intel P4

IJVM

- Implementasjon
  - Detaljar for å utføre instruksjonssettet
  - Ein gitt implementasjon har ein gitt yting
  - Endre ytinga
    - Teknologi (prosess)
    - Transistor implementasjon
    - Digital design av komponentar
    - Auke mengda logikk
    - Endring Av arkitektur
- Kva kan gjerast for å auke ytinga
  - Auke mengda logikk og endring av arkitektur



The AMD Athlon



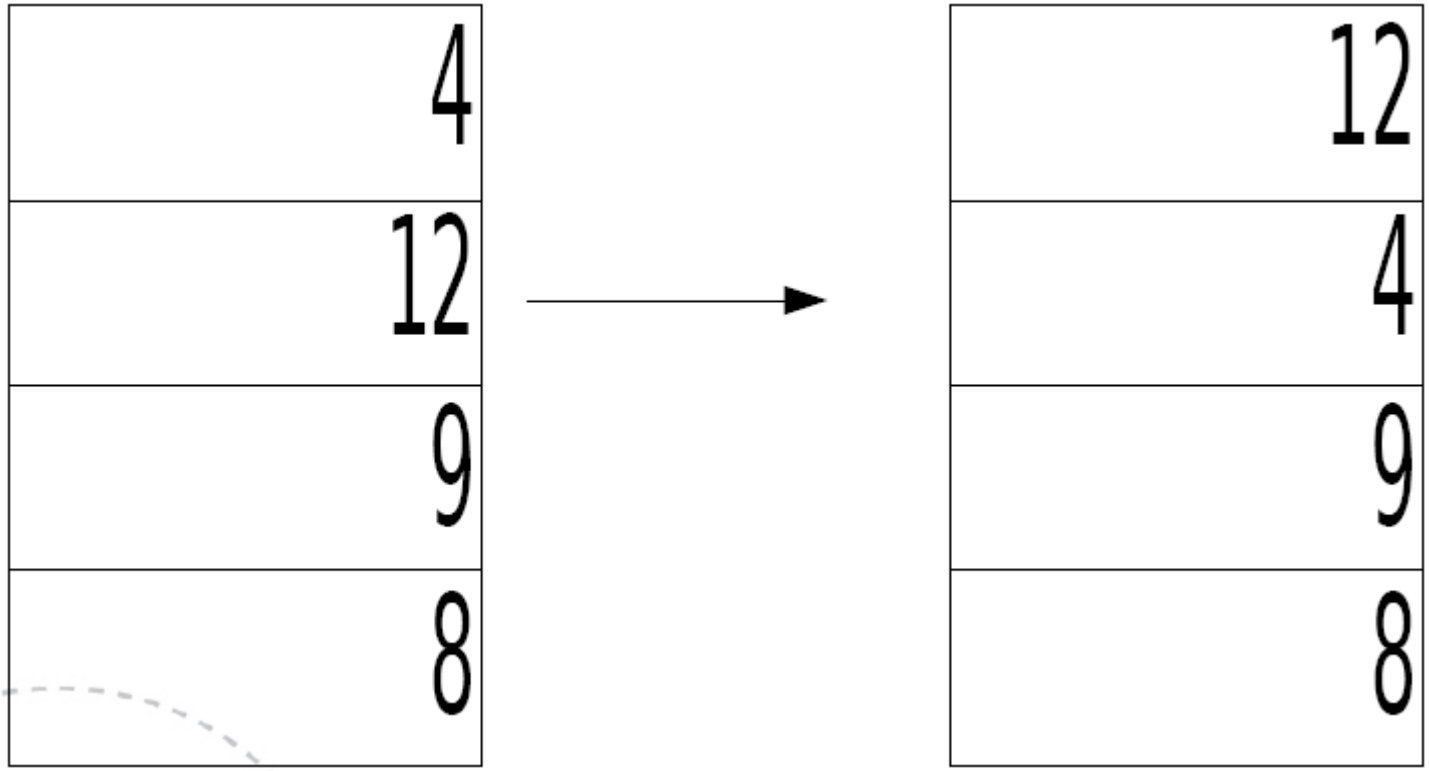
Intel P4

IJVM

- Kva kan gjerast for å auke ytinga
  - Auke mengda logikk og endring av arkitektur
    - Instruction Fetch Unit
    - Samleband
    - Instruksjonskø
    - Hurtigbuffer

Swap: opCod 0x5F

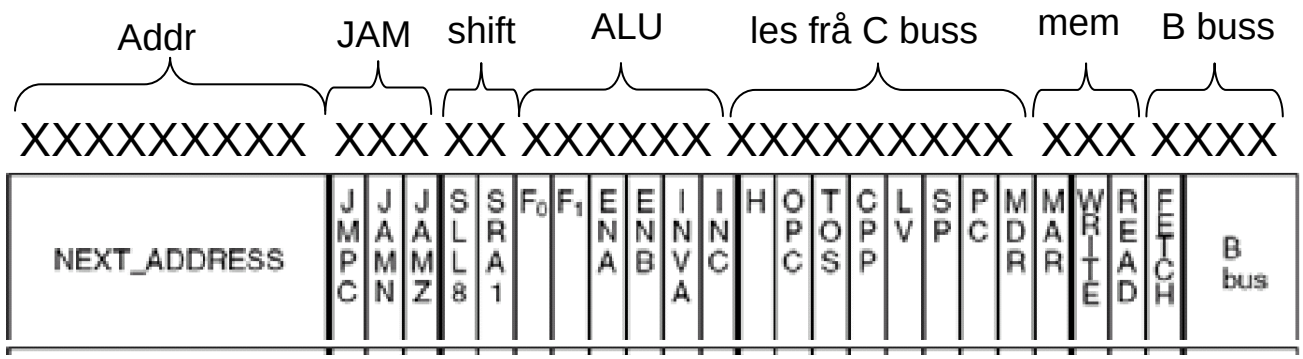
- swap1 : MAR = SP - 1, rd
- swap2 : MAR = SP
- swap3 : H = MDR, wr
- swap4 : MDR = TOS
- swap5 : MAR = SP – 1, wr
- swap6 : TOS = H; Goto Main 1

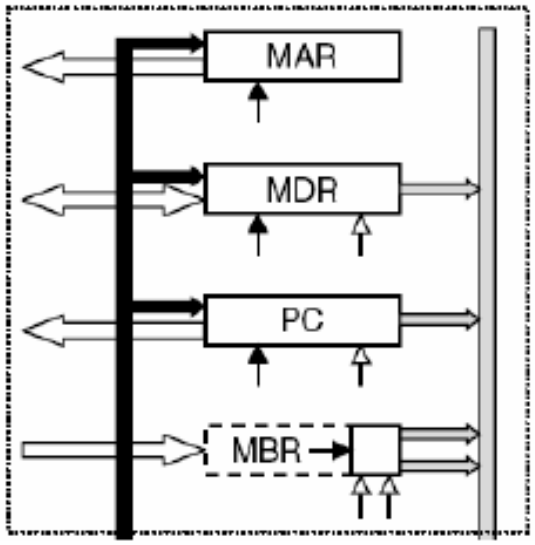


NTNU
Innovation and Creativity

Swap: opCod 0x5F

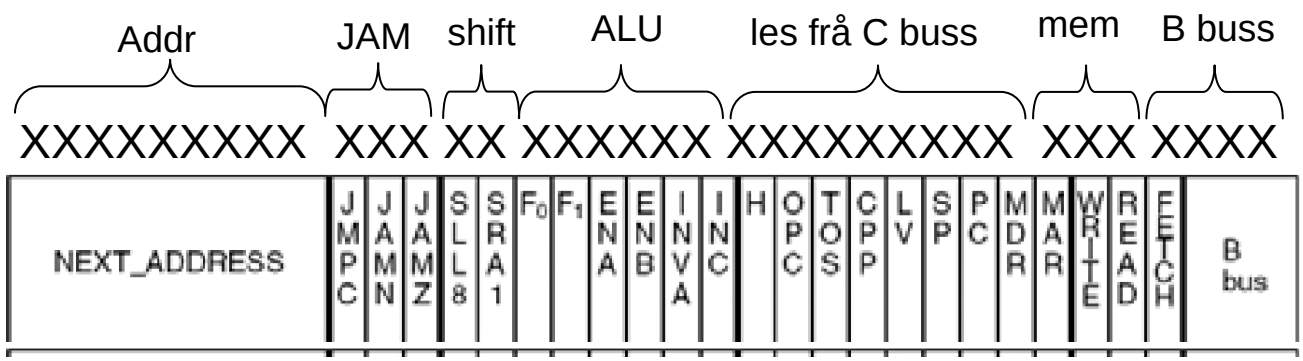
- MPC: 0x5F
- swap1 (0x5F): MAR = SP - 1, rd
  - (Addr: 0x60, JAM: 0, Shif: 0, ALU: 0x36 (b-1), cBuss: 0x01 (SP), **mem: ”010” (rd)**, bBuss: 0x4 (SP))
- swap2 (0x60): MAR = SP
- swap3 (0x61): H = MDR, wr
- swap4 (0x62): MDR = TOS
- swap5 (0x63): MAR = SP – 1, wr
- swap6 (0x00): TOS = H; Goto Main 1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

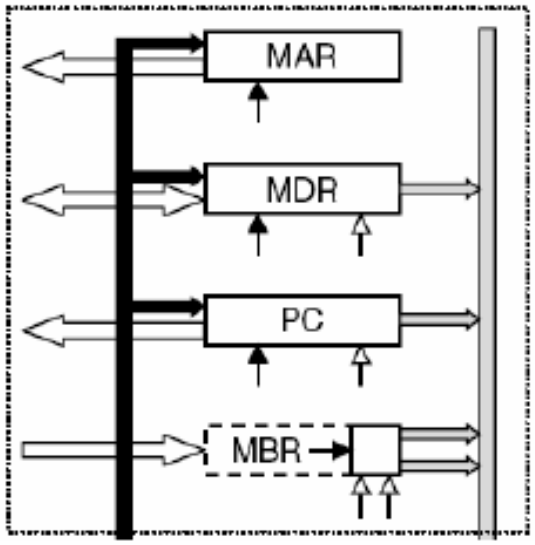




Hent neste instruksjon

- 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)



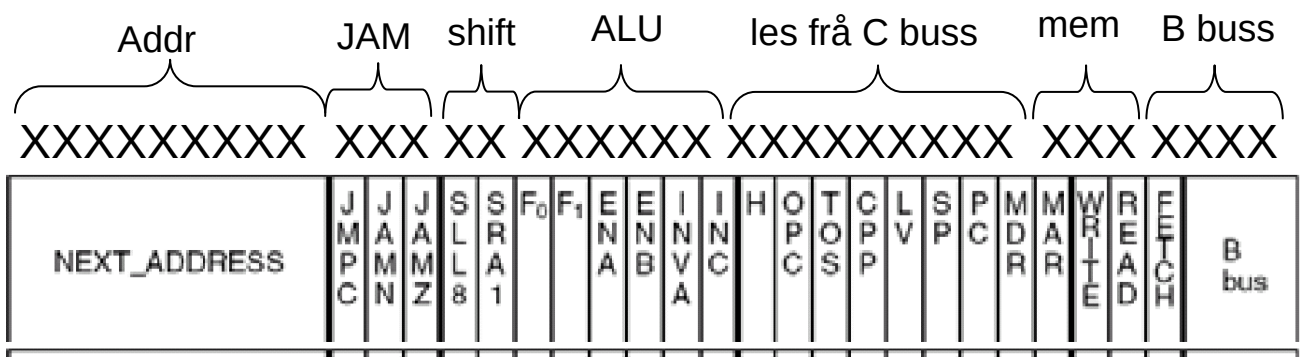


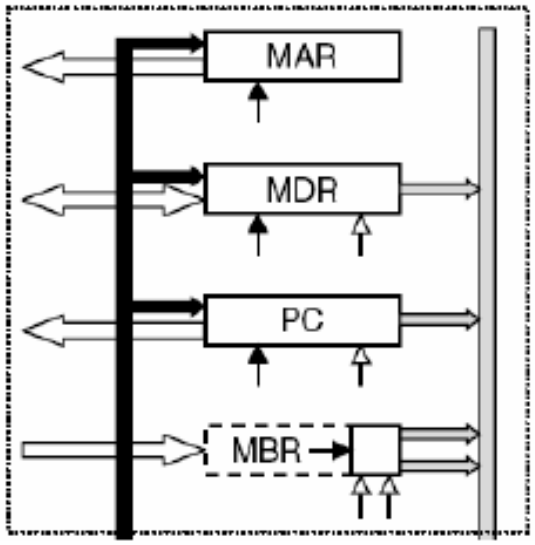
Hent neste instruksjon

- 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)





I ein liten program snutt:

```
i = j + k;
if (i == 3)
    k = 0;
else
    j = j – 1;
```

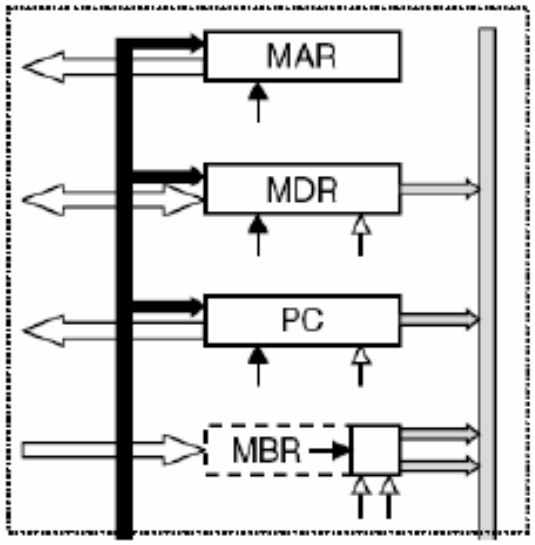
```
 1        ILOAD j          // i = j + k
 2        ILOAD k
 3        IADD
 4        ISTORE i
 5        ILOAD i          // if (i == 3)
 6        BIPUSH 3
 7        IF_ICMPEQ L1
 8        ILOAD j          // j = j – 1
 9        BIPUSH 1
10        ISUB
11        ISTORE j
12        GOTO L2
13   L1:  BIPUSH 0         // k = 0
14        ISTORE k
15   L2:
```

```
0x15 0x02
0x15 0x03
0x60
0x36 0x01
0x15 0x01
0x10 0x03
0x9F 0x00 0x0D
0x15 0x02
0x10 0x01
0x64
0x36 0x02
0xA7 0x00 0x07
0x10 0x00
0x36 0x03
```

Opkode

”Under panseret” program snutt:

- 1 ILOAD
- 2 PC = PC + 1
- 3 IADD
- 4 PC = PC + 1
- 5 ISTORE
- 6 PC = PC + 1
- 7 ILOAD
- 8 PC = PC + 1
- 9 BIPUSH
- 10 PC = PC + 1
- 11 IF_ICMPEQ (betinga hopp) **L1**
- 12 PC = Opdater PC
- 13 ILOAD
- 14 PC = PC + 1
- 15 BIPUSH
- 16 PC = PC + 1
- 17 ISUB
- 18 PC = PC + 1
- 19 ISTORE
- 20 PC = PC + 1
- 21 GOTO **L2**
- 22 PC = **L2**
- 23 BIPUSH **L1**
- 24 PC = PC + 1
- 25 ISORE
- 26 PC = PC + 1
- 27 XXXXX **L2**



”Under panseret” program snutt:

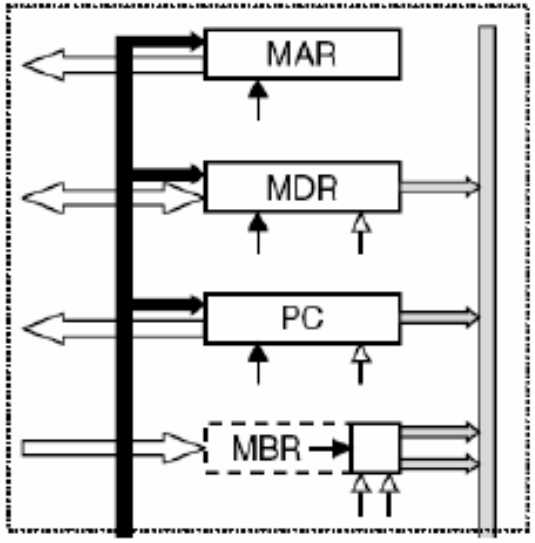
```
i = j + k;
if (i == 3)
    k = 0;
else
   j = j – 1;
```

```
 1        ILOAD j
 2        ILOAD k
 3        IADD
 4        ISTORE i
 5        ILOAD i
 6        BIPUSH 3
 7        IF_ICMPEQ L1
 8        ILOAD j
 9        BIPUSH 1
10        ISUB
11        ISTORE j
12        GOTO L2
13   L1:  BIPUSH 0
14        ISTORE k
15   L2:
```

- 1 ILOAD
- 2 PC = PC + 1
- 3 IADD
- 4 PC = PC + 1
- 5 ISTORE
- 6 PC = PC + 1
- 7 ILOAD
- 8 PC = PC + 1
- 9 BIPUSH
- 10 PC = PC + 1
- 11 IF_ICMPEQ (betinga hopp) **L1**
- 12 PC = Opdater PC
- 13 ILOAD
- 14 PC = PC + 1
- 15 BIPUSH
- 16 PC = PC + 1
- 17 ISUB
- 18 PC = PC + 1
- 19 ISTORE
- 20 PC = PC + 1
- 21 GOTO **L2**
- 22 PC = **L2**
- 23 BIPUSH **L1**
- 24 PC = PC + 1
- 25 ISORE
- 26 PC = PC + 1
- 27 XXXXX **L2**

Meir logikk høgare yting:

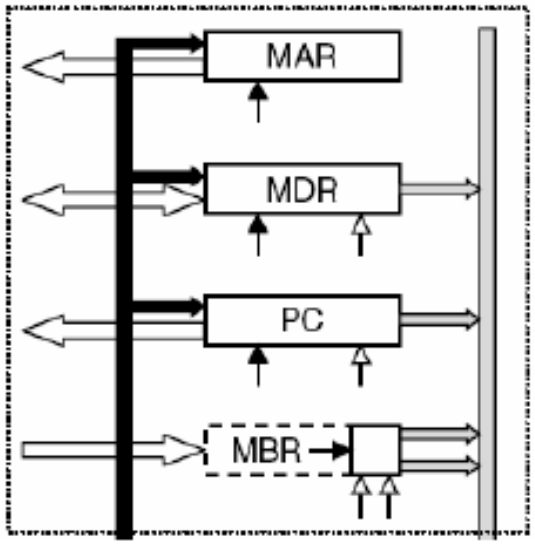
- 1 ILOAD
- 2 PC = PC + 1
- 3 IADD
- 4 PC = PC + 1
- 5 ISTORE
- 6 PC = PC + 1
- 7 ILOAD
- 8 PC = PC + 1
- 9 BIPUSH
- 10 PC = PC + 1
- 11 IF_ICMPEQ (betinga hopp) **L1**
- 12 PC = Opdater PC
- 13 ILOAD
- 14 PC = PC + 1
- 15 BIPUSH
- 16 PC = PC + 1
- 17 ISUB
- 18 PC = PC + 1
- 19 ISTORE
- 20 PC = PC + 1
- 21 GOTO **L2**
- 22 PC = **L2**
- 23 BIPUSH **L1**
- 24 PC = PC + 1
- 25 ISORE
- 26 PC = PC + 1
- 27 XXXXX **L2**



Flytte funksjonalitet til maskinvare:

- 1 ILOAD
- 2 PC = PC + 1
- 3 IADD
- 4 PC = PC + 1
- 5 ISTORE
- 6 PC = PC + 1
- 7 ILOAD
- 8 PC = PC + 1
- 9 BIPUSH
- 10 PC = PC + 1
- 11 IF_ICMPEQ (betinga hopp) **L1**
- 12 PC = Opdater PC
- 13 ILOAD
- 14 PC = PC + 1
- 15 BIPUSH
- 16 PC = PC + 1
- 17 ISUB
- 18 PC = PC + 1
- 19 ISTORE
- 20 PC = PC + 1
- 21 GOTO **L2**
- 22 PC = **L2**
- 23 BIPUSH **L1**
- 24 PC = PC + 1
- 25 ISORE
- 26 PC = PC + 1
- 27 XXXXX **L2**

- PC = PC +1
  - **ALU**
  - **B-Buss**
  - **C-Buss**
- Okuperar ressursar

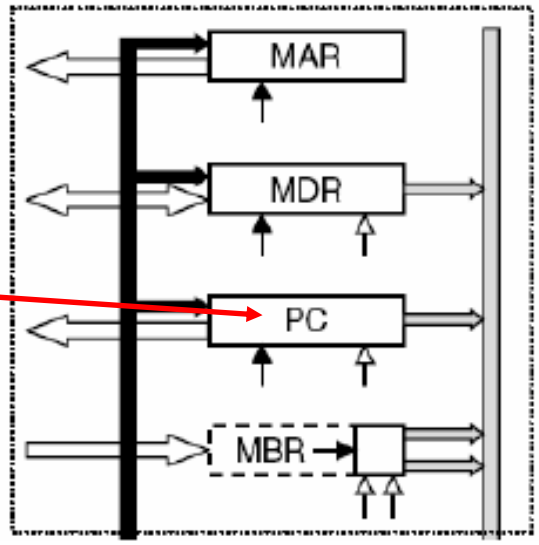


Flytte funksjonalitet til maskinvare:

- 1 ILOAD
- 2 PC = PC + 1
- 3 IADD
- 4 PC = PC + 1
- 5 ISTORE
- 6 PC = PC + 1
- 7 ILOAD
- 8 PC = PC + 1
- 9 BIPUSH
- 10 PC = PC + 1
- 11 IF_ICMPEQ (betinga hopp) **L1**
- 12 PC = Opdater PC
- 13 ILOAD
- 14 PC = PC + 1
- 15 BIPUSH
- 16 PC = PC + 1
- 17 ISUB
- 18 PC = PC + 1
- 19 ISTORE
- 20 PC = PC + 1
- 21 GOTO **L2**
- 22 PC = **L2**
- 23 BIPUSH **L1**
- 24 PC = PC + 1
- 25 ISORE
- 26 PC = PC + 1
- 27 XXXXX **L2**

- PC = PC +1
  - **ALU**
  - **B-Buss**
  - **C-Buss**
- Okuperar ressursar
  - Legg til logikk
    - PC = PC + 1

•PC = PC +1
•Hopp

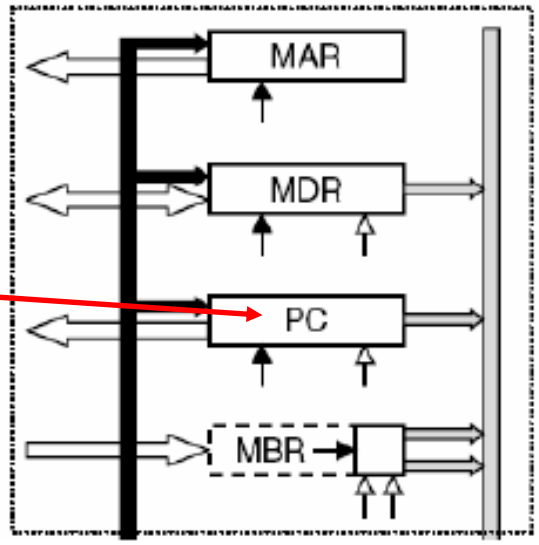


Flytte funksjonalitet til maskinvare:

- 1 ILOAD
- 2 IADD
- 3 ISTORE
- 4 ILOAD
- 5 BIPUSH
- 6 IF_ICMPEQ (betinga hopp) **L1**
- 7 PC = Opdater PC
- 8 ILOAD
- 9 BIPUSH
- 10 ISUB
- 11 ISTORE
- 12 GOTO **L2**
- 13 PC = **L2**
- 14 BIPUSH **L1**
- 15 ISORE
- 16 XXXXX **L2**

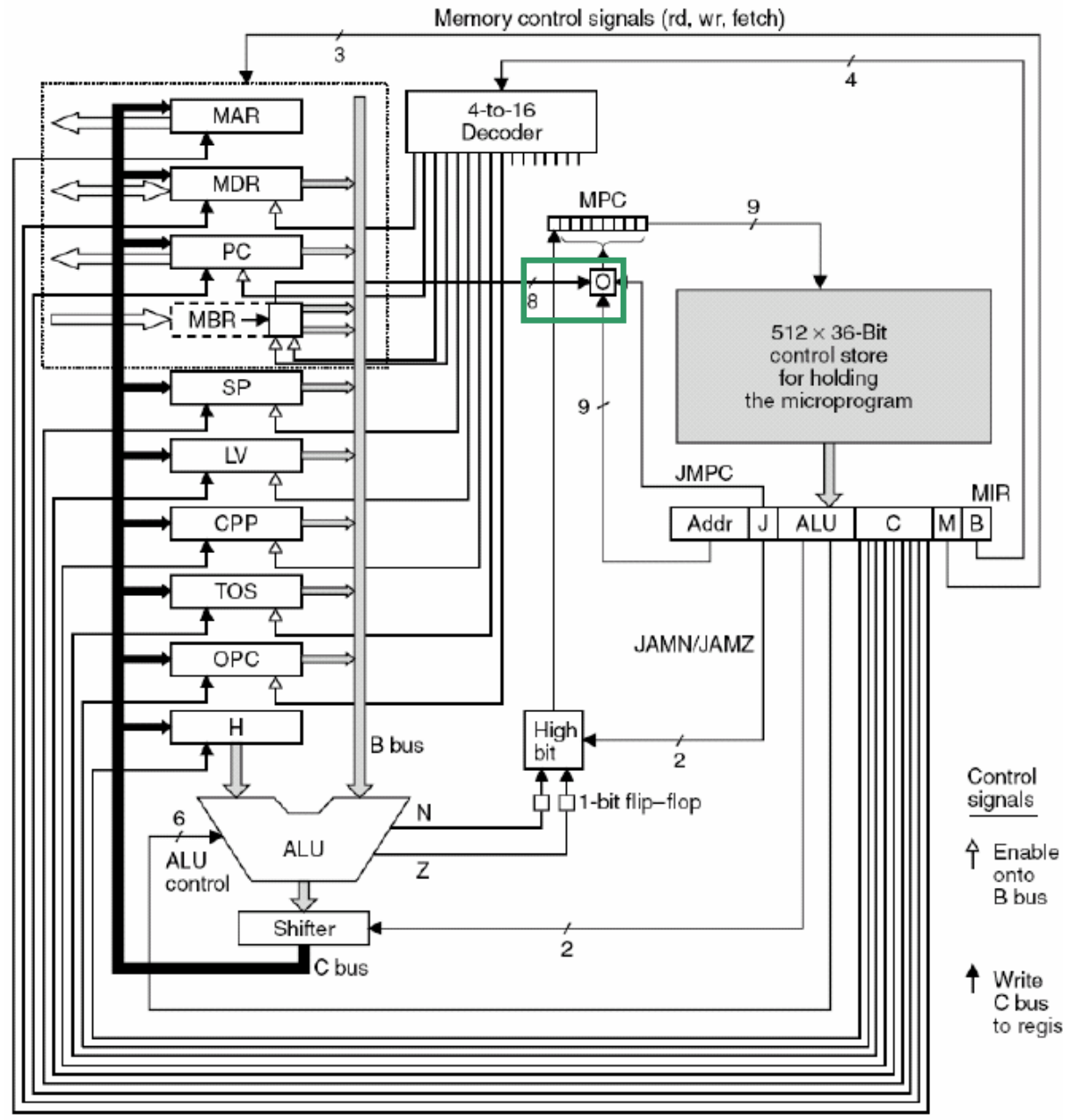
- PC = PC +1
  - **ALU**
  - **B-Buss**
  - **C-Buss**
- Okuperar ressursar
  - Legg til logikk
    - PC = PC + 1
  - Redusert med 11 gonger bruk av
    - ALU
    - B-Buss
    - C-Buss

•PC = PC +1
•Hopp



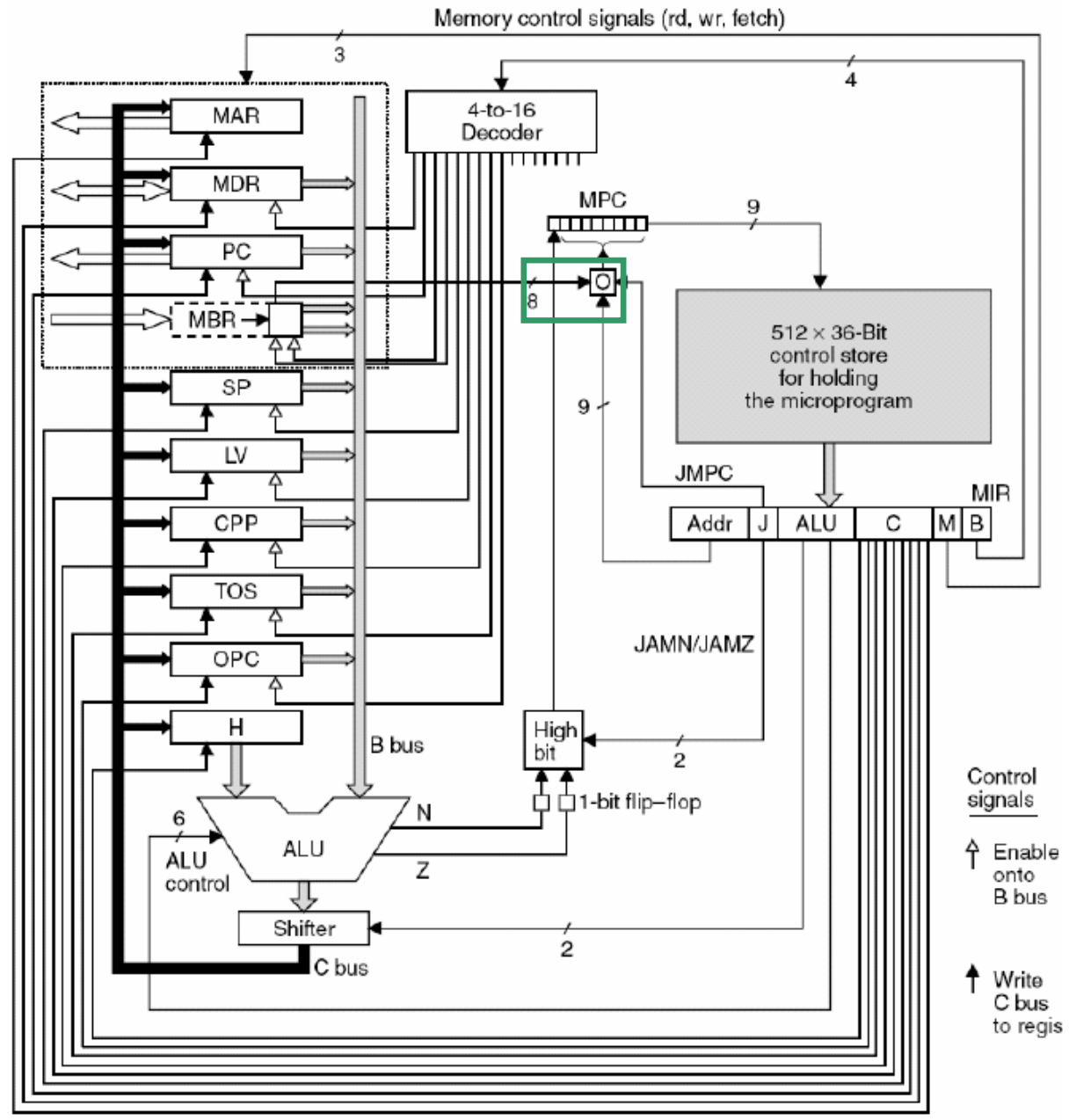
IJVM

- Instruksjon
  - 8 bit (1 byte)
- Minne grensesnitt
  - 32 bit (4 Byte)



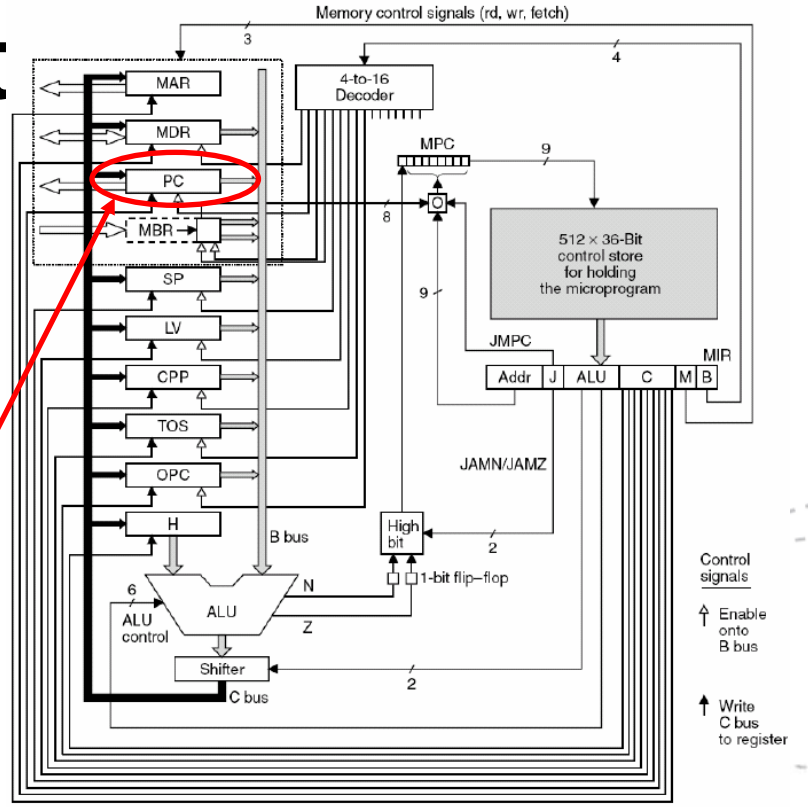
IJVM

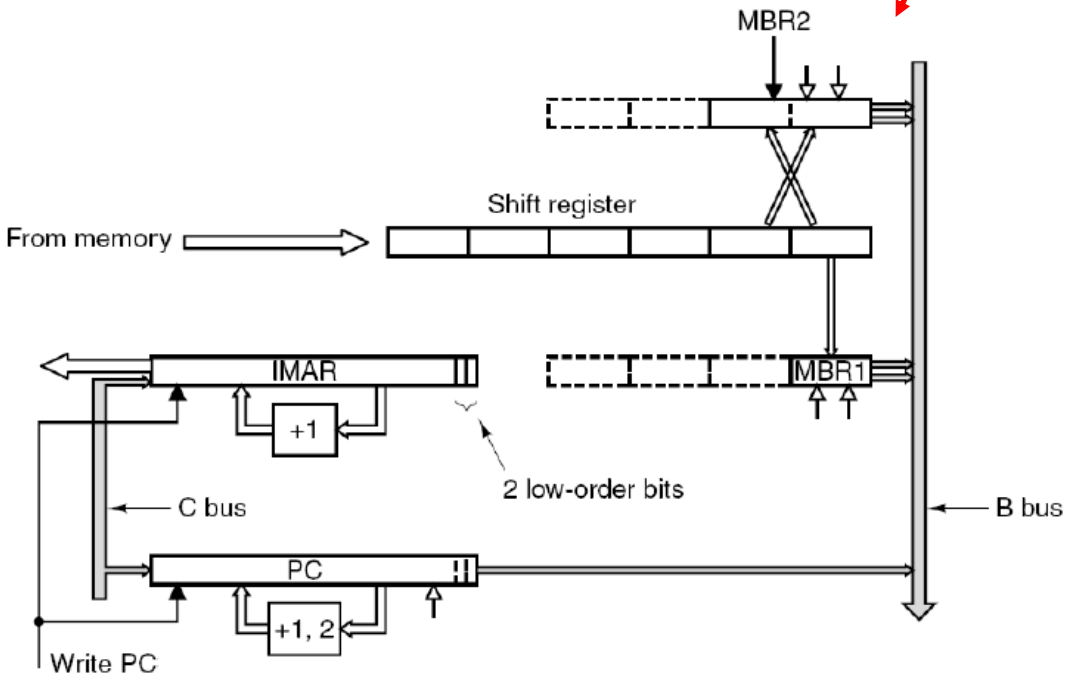
- Instruksjon
  - 8 bit (1 byte)
- Minne grensesnitt
  - 32 bit (4 Byte)
- Brukar
  - 32 bit buss
  - Hentar 8 bit
  - 24 bit ubrukt



Instruction Fetch Unit

- Instruksjon
  - 8 bit (1 byte)
- Minne grensesnitt
  - 32 bit (4 Byte)
- Brukar
  - 32 bit buss
  - Hentar 8 bit
  - 24 bit ubrukt





NTNU
Innovation and Creativity