Velkommen til siste gruppetime!

Repetisjon / eksamensforberedelse

Gruppetime 13: 23. november 2021

(Hanne-)Kristin Schrøder Leiros

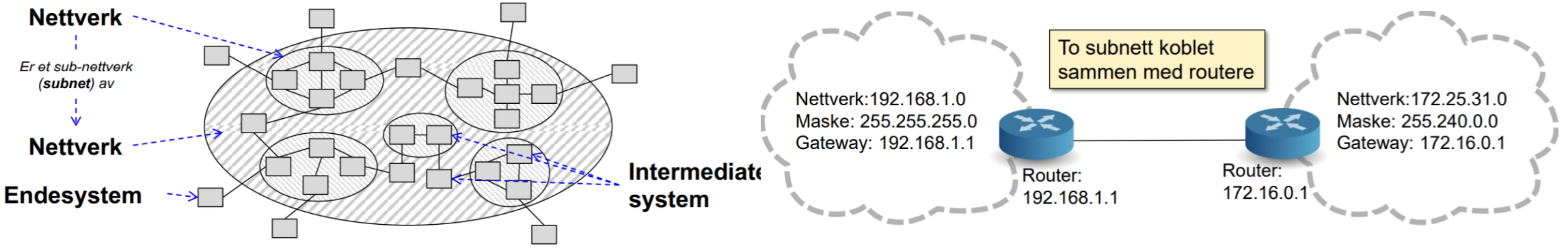
hkleiros@uio.no

Plan for dagen

- Repetisjon av spørsmål fra dere
  inkl noen ukesoppgaver om det er ønskelig å gjennomgå
  - IP-adresser
  - UDP og TCP
  - Utregning av frekvens og cache-miss
  - LMC
- Tidligere eksamensoppgaver
  - LMC H2019
  - LMC og digital representasjon H2018
  - * Har alle tidligere poller om det er ønsker om å gjøre noen igjen ☺
- Som alltid bare å stille spørsmål eller avbryte om jeg skal utdype / repetere underveis

IP-adresse

- «Unik» adresse som tildeles en enhet i et TCP/IP-basert datanettverk
  - Statiske som bedrifter ofte benytter og dynamiske som tildeles av internettselskap
- Brukes for å koble sammen enheter og vite hvor man skal sende nettverkspakker
- Internett består av mange subnett som fremstår som én enhet



Endesystemer (tjenere og klienter):

- *For eksempler:*
  - Datamaskiner
  - Mobiltelefoner
  - «Duppedingser» (IoT), skrivere, etc.

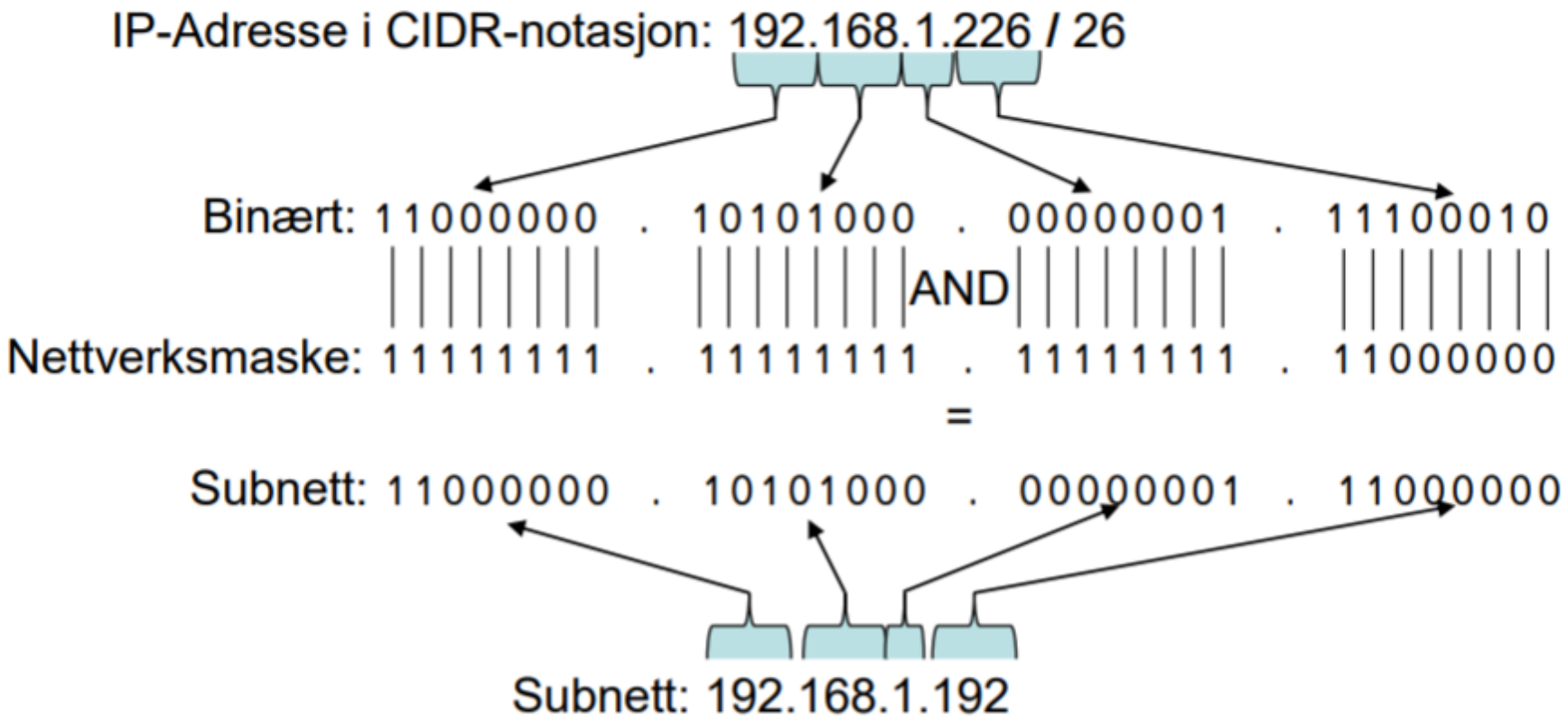
Intermediate system (rutere og switcher):

- *For eksempel:*
  - Ruter
  - Switch

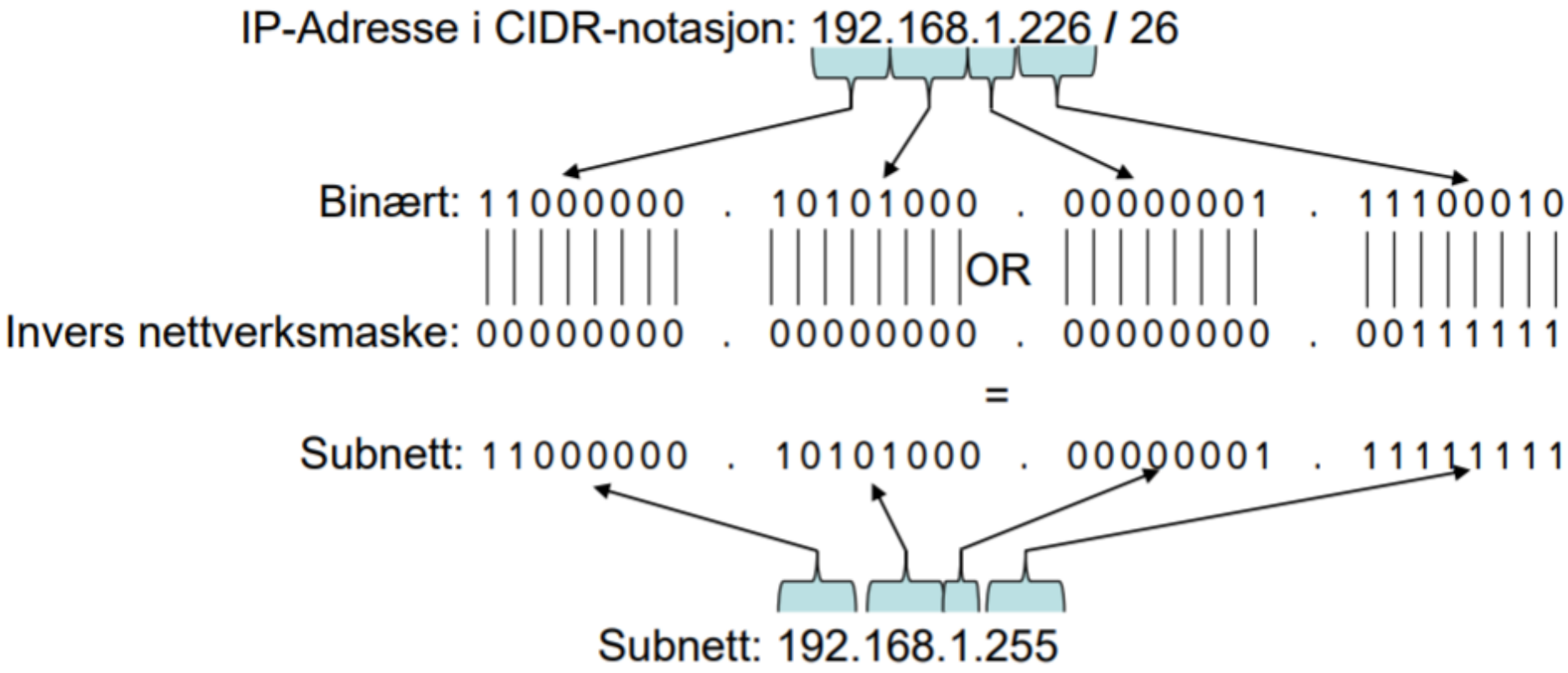
Utregning av IP-adresser

- En IPv4-adresse består av 4 byte, eller oktetter, der hver har maksverdi 255
- Nettverksmaske bestemmer hvilke bits som definerer et subnett
  - 11111111.11111111.11111111.00000000
  - 255.255.255.0 –gir 255 IP-adresser i subnettet der 253 kan deles ut
- For å finne subnettadressen til et nettverk, utfører man en binær AND operasjon med IP-adressen og nettverksmasken
  - I praksis betyr dette at den blir lik der det er 1-ere og 0 der det er 0-ere i nettverksmasken
  - Se neste slide
- For å finne kringkastingsadressen, utfører man en binær OR operasjon med IP-adressen og INVERSEN av nettverksmasken
  - I praksis betyr dette at adressen blir lik IP-frem til det ville kommet 0-ere i nettverksmasken
  - Se to slides frem

Eksempel: subnettadresse fra IP / nettverksmaske



Eksempel: kringkastingsadresse fra IP / nettverksmaske

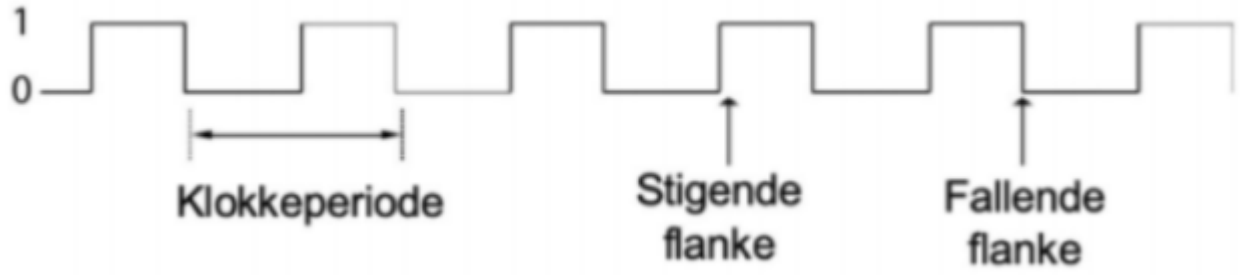


Hva er en IP-adresse?

Ukesoppgave IP-adresser: 44.2

Klokkesignaler (7. okt)

- Fordi det er fysiske begrensninger på transistorer vil det oppstå forsinkelse i kretser
- For å kunne synkronisere større kretser, slik at ingen porter må stå å vente en uvvis tid på resultater brukes klokkesignaler.
- Disse består av klokkeperioder som måles fra en stigende eller fallende flanke til neste stigende eller fallende flanke.
- Når signalet er 1 utføres operasjoner, mens når det er 0 venter komponentene og man forsikrer seg om at hvert signal sendes til rett tid
- Frekvens beskriver hvor fort klokken endrer signal
- Jo høyere frekvens klokkesignalet har, jo raskere kan en datamaskin fungere men dette er igjen begrenset at fysikken



Utregning av frekvens

- https://www.sensorsone.com/period-to-frequency-calculator/
- Formelen for Picosekunder til GigaHertz
  - 1 / ps * 1000 = GHZ
- **Ukesoppgave 27-30 uke 41**

Utregning av cache-miss

- N = total antall instruksjoner
- % miss = minneakksesering*cache-miss
- Om penaly:
  - N * standard tid + (N * %miss * ekstra tid)
- Om «cache-miss fører til total tidsbruk»:
  - (N – N*minneaksessering)*standard tid + (N*miss%*total tid)
- **U 42: oppgave 3 - *Scenario 2***
- **H2019: 4.3 - *Scenario 2***

- Scenario 1: cache-miss har penalty

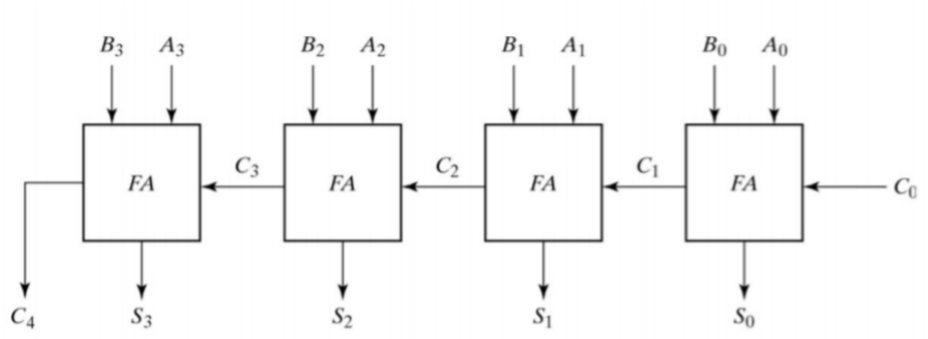
Total antall instruksjoner * tid per instruksjon + ((antall instruksjoner som trenger mineaksessering * prosent cache-miss)*penalty) = total tidsbruk

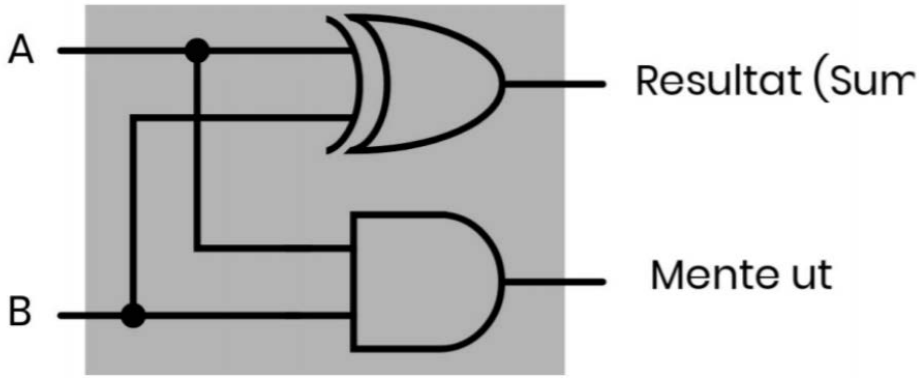
- Scenario 2: cache-miss fører til total tidsbruk (for den instruksjonen)

(Total antall instruksjoner – antall instruksjoner som trenger minnekasessering) * tid per instruksjon
+ (antall instruksjoner som trenger minneaksesering * prosent cache-miss) * total tid ved cache-miss) = total tidsbruk

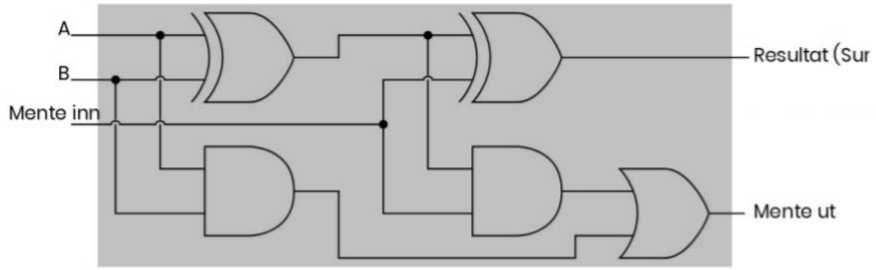
Binær addisjon

- Skjer, som desimal addisjon, ved å legge sammen tall og føre mente
- Kan utføres med en halv-adder
- Men om vi også har mente inn trenger vi en full-adder
- Men disse adderer bare et bit så for å kunne addere flere bit trenger vi serielle addere
  - Altså flere full-addere (FA) koblet sammen





HALV-ADDER (HA)



FULL-ADDER (FA)

https://www.uio.no/studier/emner/matnat/ifi/IN1020/h21/undervisningsmateriale/02---studentnotater-2021.pdf

TCP – Transmission Control Protocol

- Mest brukte protokollen på Transportlaget i TCP/IP-modellen
- Forbindelsesorientert
  - Setter opp et 3-veis-håndtrykk, der hver pakke bekreftes mottatt og det bekreftes av det er bekreftet
    - Pålitelighet
- Flytkontroll
  - Sender ikke informasjon fortere enn mottakeren kan ta imot
- Levering i riktig rekkefølge
- Feilsjekking av nyttelasten (sjekksum)
  - Bruker en Hash-algoritme til å sjekke at meldingen ikke er endret under sendingen
- Mentingskontroll
  - Sender ikke pakker fortere enn ruteren kan håndtere

UDP – User Datagram Protocol

- «Minimal meldingsorientert nettverksprotokoll for forbindelsesløs overføring av informasjon»
- Tilkoblingsløs – ingen holder oversikt over status til forbindelsen
- Ingen garantier for at pakker kommer frem
  - Sender med «best effort» prinsipp
- Feilsjekking av nyttelasten (sjekksum)

Ukesoppgaver protokoller: uke 43, 44.1

Instruksjoner – assemblerkode og tall

- HLT – 000
  - Stopper kjøringen
- INP – 901
  - Ber brukeren skrive et tall
  - Legger tallet i akkumulatorregisteret
- OUT – 902
  - Skriver ut verdien i akkumulatoren
- STA – 3xx
  - «Store accumulator»
  - Kopier verdien i akkumulatoren til adresse xx i minnet
- LDA – 5xx
  - «Load accumulator»
  - Kopierer verdien i lokasjon xx til akkumulatoren
- ADD – 1xx
  - Adderer verdien i adresse xx i minnet til akkumulatoren
- SUB – 2xx
  - Trekker verdien i adresse xx i minnet fra akkumulatoren
- BRA – 6xx
  - «Branch»
  - Hopper til adresse xx
- BRP – 8xx
  - «Branch if positive»
  - Hopper til adresse xx HVIS akkumulatoren inneholder en verdi som er >=0
- BRZ – 7xx
  - «Branch if zero»
  - Hopper til adresse xx HVIS akkumulatoren inneholder verdien 0,

Kode	Navn	Beskrivelse
0xx	HLT	Stopper eksekveringen
1xx	ADD	Adderer verdien i angitt minnelokasjon med akkumulatoren
2xx	SUB	Subtraherer verdien i angitt minnelokasjon med akkumulatoren
3xx	STA	Lagrer akkumulatoren i angitt minnelokasjon
4xx	-	Ikke i bruk
5xx	LDA	Henter verdi fra minnet til akkumulatoren
6xx	BRA	Hopper til angitt adresse
7xx	BRZ	Hopper hvis akkumulatoren er 0
8xx	BRP	Hopper hvis akkumulatoren er ≥ 0
901	INP	Leser verdi fra input, og legger svaret i akkumulatoren
902	OUT	Skriver ut verdien i akkumulatoren
922	OTC	Skriver ut ASCII-tegn (ikke i boka)

Eksamensoppgaver LMC og litt digital representasjon

H2019

Assemblerkode 1

```
start   INP
        STA     x
        INP
        STA     y
        LDA     x
        SUB     y
        OUT
slutt   HLT

x       DAT     0
y       DAT     0
```

Hva skrives ut når denne koden kjøres og brukeren gir **7** og **11** som input?

Assemblerkode 2

```
start   INP
        BRZ     b
        LDA     x
        ADD     y
        STA     x
        BRA     start

b       LDA     x
        OUT
slutt   HLT

x       DAT     0
y       DAT     1
```

Hva skrives ut når dette programmet kjøres og brukeren gir verdiene **5 3 3 1 0** som input?

Assemblerkode 3

```
start   INP
        ADD     b
        STA     b
b       LDA     u
        OTC
        HLT

u       DAT     83      # ASCII 'S'
        DAT     77      # ASCII 'M'
        DAT     84      # ASCII 'T'
        DAT     79      # ASCII 'O'
        DAT     84      # ASCII 'T'
        DAT     70      # ASCII 'F'
        DAT     76      # ASCII 'L'
```

Hva skrives ut i dette programmet når bruker gir **6** som input?

Maskinkode

Anta at minnet i LMC-maskinen inneholder disse verdiene

0	**901**
1	**310**
2	**110**
3	**311**
4	**111**
5	**311**
6	**111**
7	**210**
8	**902**
9	**0**
10	**0**
11	**0**

og at programtelleren («Program counter») er **0**. Hva skrives ut når koden kjøres og brukeren angir **8** som input?

H2018

Binære tall 2

En byte inneholder disse bit-ene:

1	1	0	0	1	0	0	0

Hvilke verdier kan representeres av disse bit-ene?

Maskinkode 1

```
start   INP
        STA   50
        ADD   50
        ADD   50
        OUT
slutt   HLT
```

Hva skrives ut når denne koden kjøres og brukeren gir **7** som inndata?

Maskinkode 3

Anta at minnet inneholder disse verdiene:

0:	901
1:	805
2:	508
3:	902
4:	000
5:	108
6:	308
7:	600
8:	000

Anta også at *programtelleren* (også kalt «program counter» eller «location counter») inneholder **0**. Hva blir da skrevet ut når vi starter prosessoren vår og brukeren gir verdiene **10**, **1** og **-2** som inndata?

Maskinkode 4

```
start   INP
        BRZ     slutt
        BRP     k
        BRA     start
k
         :
        BRA     start
slutt   LDA     x
        OUT
        HLT

x       DAT     0
y       DAT     1
```

```
a       LDA     x
        ADD     1
        STA     x
```

```
b       LDA     1
        ADD     x
```

```
c       LDA     x
        ADD     y
        STA     x
```

```
d       ADD     x
        STA     x
```

```
e       ADD     y
        STA     x
```

- Dette programmet (i den tykke røde rammen) skal lese inn divserse tall. Det siste tallet skal være 0, og det er ingen andre forekomster av 0. Programmet skal telle antall ekte positive tall (dvs tall som er >0) og skrive ut dette antallet. Negative tall skal ignoreres.
- Sekvensen
  - **3 3 4 -1 -2 5 0**
  - Skal gi svaret 4 siden det er 4 tall som er > 0
- Programmet mangler noen instruksjoner der det står et kolon (:). Hvilke instruksjoner mangler?