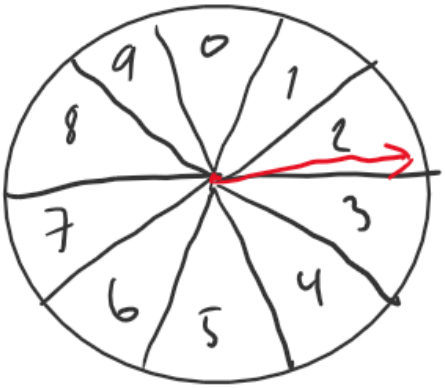
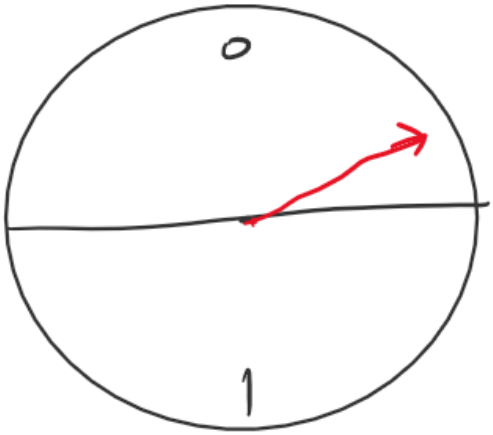
1a.

Datamaskiner bruker binær logikk fordi det er det som blitt standarden. Man kunne ha for eksempel brukt base10-logikk.

Grunnen til at man gikk for binær var fordi det er pålitelig, da minker sannsynligheten for signalfeil – Det er enten av eller på. Mens med base10 er det større rom for feil med sine 10. Tegningene illustrere det:





En annen grunn av på grunn av det er billigere å produsere «enkle» kretser sammenlignet med mer komplekse kretser.

1b)

TCP/IP modellen:

1. Applikasjonslaget
Bekskriver hvordan applikasjonene kommuniserer mellom to maskiner
Kan brukes til å web browsing, laste ned filer og sende epost.
Eksempler på protokoller/ standarder: HTTP, SMTP, FTP

2. Transportlaget
Lager og opprettholder forbindelse mellom to maskiner, eks handshake, samt
Eksempler på protokoller/standarder: TCP og UDP

3. Nettverkslaget
Håndterer IP og routing mellom maskinene
Eksempler på protokoller/standarder: IPv4, IPv6, ICMP

4. Link-laget
Har ansvaret for transport mellom noder (switching)
Eksempler på protokoller: Ethernet, FDDI, IEEE 802.11

5. Fysiske laget
Fysisk tranport av dataene (bits). Kabel/fiber og WIFI
Eksempler på protokoller/standarder: 10base-T, 100base-T, 1000base-T

1c)

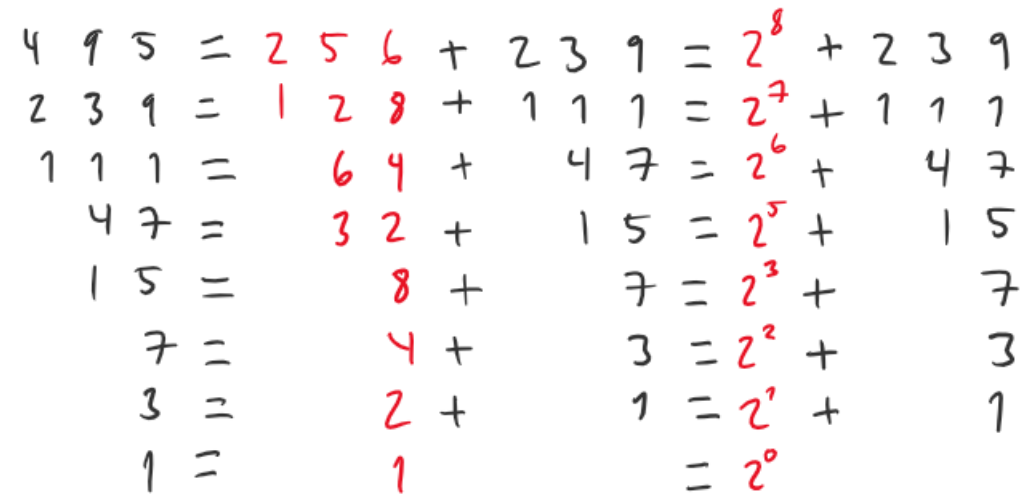
Et operativsystem er et sett av programmer som ligger mellom hardwaren og andre programmer. Den har som oppgave å tildele, regulere og fordele hardware-ressurser, som for eksempel CPU og minne, til ulike programmer.

Forskjellen på kernel mode og user mode, er at kernel mode har full tilgang til hardware-ressursene, mens user mode har begrenset tilgang.

2a)

Konvertering fra desimal til binær med 16 bits presisjon:

$495_{10} = ?_2$



$55_{10} = ?_2$

$55 = 32 + 23 = 2^5 + 23$

$23 = 16 + 7 = 2^4 + 7$

$7 = 4 + 3 = 2^2 + 3$

$3 = 2 + 1 = 2^1 + 1$

$1 = 1 = 1^0$

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1

$55_{16} = 0000000000110111_2$

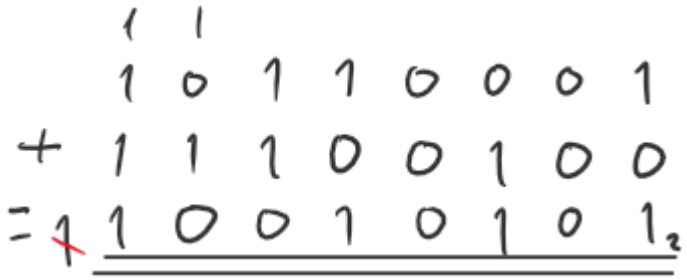
2b)

Binær addisjon med 8 bits presisjon:

$0111\ 0101_2 + 0010\ 0110_2 = ?_2$

	1	1			1			
	0	1	1	1	0	1	0	1
+	0	0	1	0	0	1	1	0
=	1	0	0	1	1	0	1	1_2

$1011\ 0001_2 + 1110\ 0100_2 = ?_2$

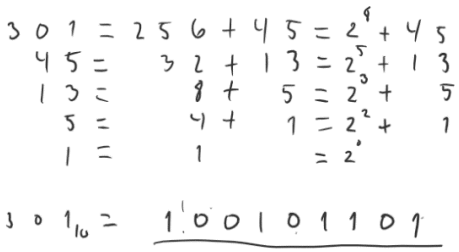


2c)

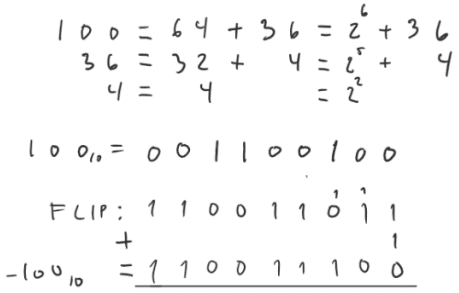
Binær subtraksjon:

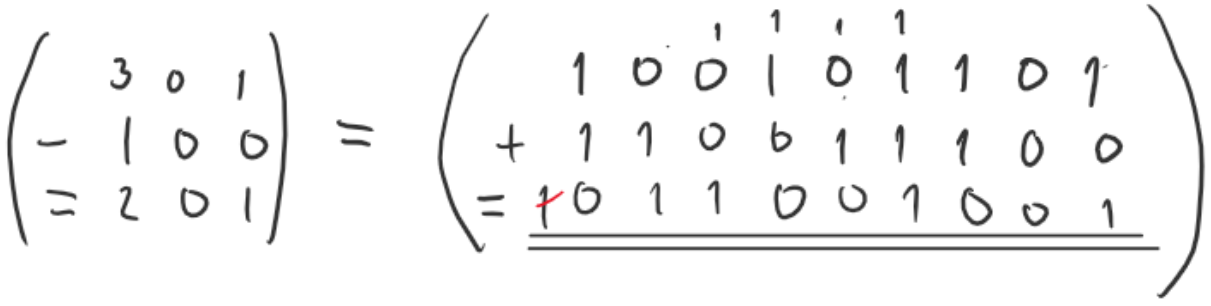
$301_{10} - 100_{10} = ?_2$

301 i binær



-100 i binær





2d)

Unicode består av kodepunkter for hvert tegn, fordelt på 17 plan hvor hvert plan består av 65 536 kodepunkter. Unicode kan enkodes på forskjellige måter, for eksempel ASCII, UTF-8, UTF-16 og UTF-32.

2e)

UTF-8 enkoding av Unicode-punktet U+2655

Omregne 0x2655 til binær: 0010 0110 0101 0101

Fylle på vognene (16 biter): **1110**0010 **10**011001 **10**010101

Omregne binær til hex: 0xE29995

2f)

Boolske operasjoner

Input		**Output**
0x42	**0xF2**	**0x42 & 0xF2**
0	1	**0**
1	1	**1**
0	1	**0**
0	1	**0**
0	0	**0**
0	0	**0**
1	1	**1**
0	0	**0**

Svar: 0x42 & 0xF2 = 0100 0010 = 0x42

Input		**Output**
0x39	**0xAC**	**0x39 & 0xAC**
0	1	**0**
0	0	**0**
1	1	**1**
1	0	**0**
1	1	**1**
0	1	**0**
0	0	**0**

1	0	**0**

Svar: 0x39 & 0xAC = 0010 1000 = 0x28

Input		**Output**
0x55	0x72	**0x55 | 0x72**
0	0	**0**
1	1	**1**
0	1	**1**
1	1	**1**
0	0	**0**
1	0	**1**
0	1	**1**
1	0	**1**

Svar: 0x55 | 0x72 = 0111 0111 = 0x77

Input		**Output**
Input A: 0xFF	**Input B: 0x34**	**0xFF | 0x34**
1	0	**1**
1	0	**1**
1	1	**1**
1	1	**1**
1	0	**1**
1	1	**1**
1	0	**1**
1	0	**1**

Svar: 0xFF | 0x34 = 1111 1111 = 0xFF

Input		**Output**
Input A: 0x69	**Input B: 0xBB**	**0x69 ^ 0xBB**
0	1	**1**
1	0	**1**
1	1	**0**
0	1	**1**
1	1	**0**
0	0	**0**
0	1	**1**
1	1	**0**

Svar: 0x69 ^ 0xBB = 1101 0010 = 0xD2

2g)

konverter fra base21 til base10:

Base 21	1	8	1	4

	21^3	**21^2**	**21^1**	**21^0**
Base 10	(21^3) * 1	(21^2) * 8	(21^1) * 1	(21^0) * 4
=	9261 * 1	+ 441 * 8	+ 21 * 1	+ 1 * 4
=	9 261	+ 3528	+ 21	+ 4
=	12 814			

3a)

kommandoen "**ping www.h-ck.me > list**" på kommandolinje gjør følgende:

- **"ping www.h-ck.me**" sender 4 ICMP-pakker(forespørsel) til serveradressen. Hvis serveren mottar pakken, svarer den med 4 ICMP-pakker (svar på forespørsel). På denne måten vet om det er forbindelse med mellom avsender- og mottakermaskin. Vi får også informasjon:
    - Om hvor lang tid det tok fra vi sendte pakkene til vi fikk svar retur
    - Om noen av pakkene har gått tapt
    - IP-adressen til serveren
- **«> list»** outputer resultatet vi får fra "**ping www.h-ck.me**" til fil med navn «**list**» uten angitt filformat. Hvis det ikke finnes en fil med navnet «**list**» i output-mappen, opprettes det en ny fil, og hvis det finnes fra før, overskrives denne.

list – Notisblokk

Fil Rediger Format Vis Hjelp

```
Pinging www.h-ck.me [46.30.215.126] with 32 bytes of data:
Reply from 46.30.215.126: bytes=32 time=9ms TTL=50
Reply from 46.30.215.126: bytes=32 time=9ms TTL=50
Reply from 46.30.215.126: bytes=32 time=9ms TTL=50
Reply from 46.30.215.126: bytes=32 time=9ms TTL=50

Ping statistics for 46.30.215.126:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 9ms, Maximum = 9ms, Average = 9ms
```

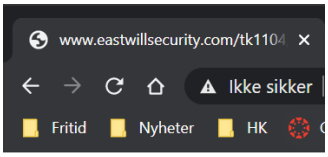
(Output: åpnet filen i .txt-format)

3b)

For å se maskinens routing-table på kommandolinjen, skriver man kommandoen «**route print**»

3c)

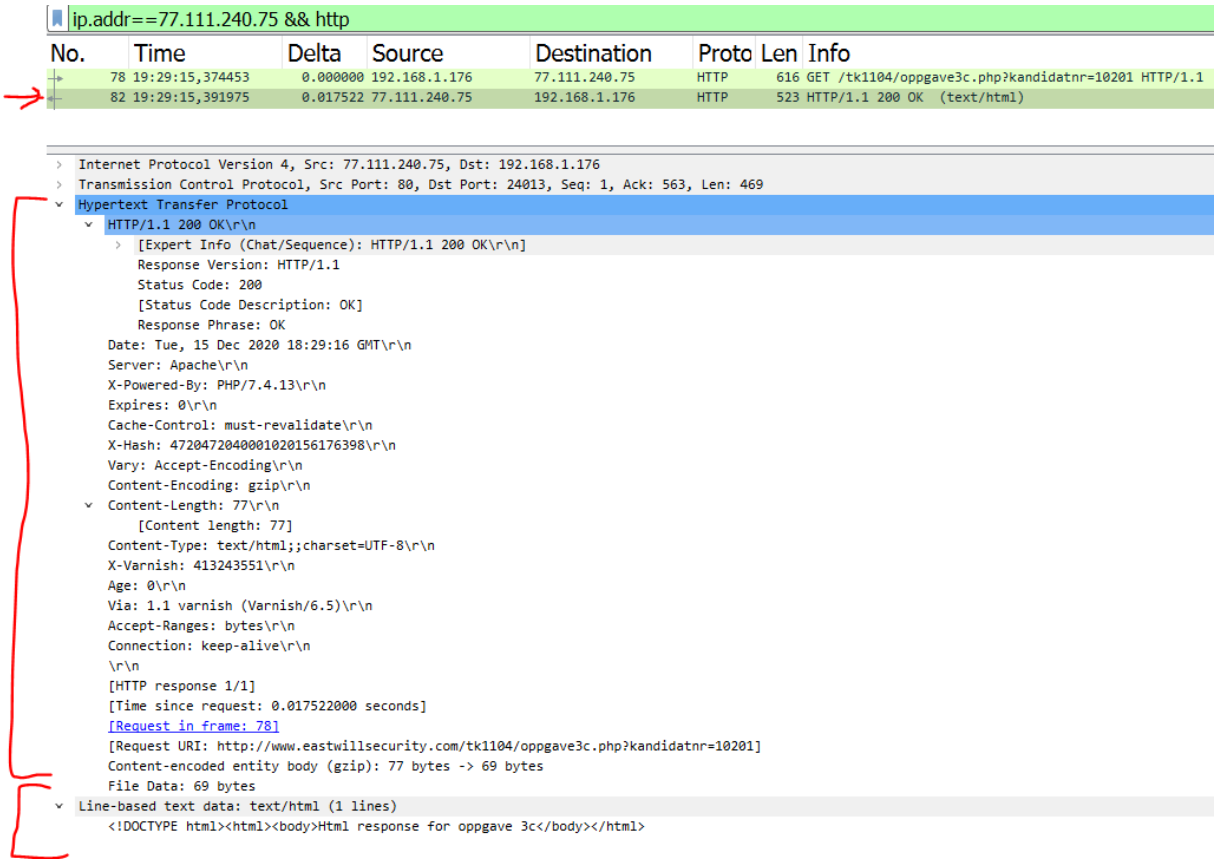
1. Åpnet Wireshark og startet opptak av nettverkstrafikken(wifi)
2. Åpnet Chrome-nettleser og la inn URL: http://www.eastwillsecurity.com/tk1104/oppgave3c.php?kandidatnr=10201



Html response for oppgave 3c

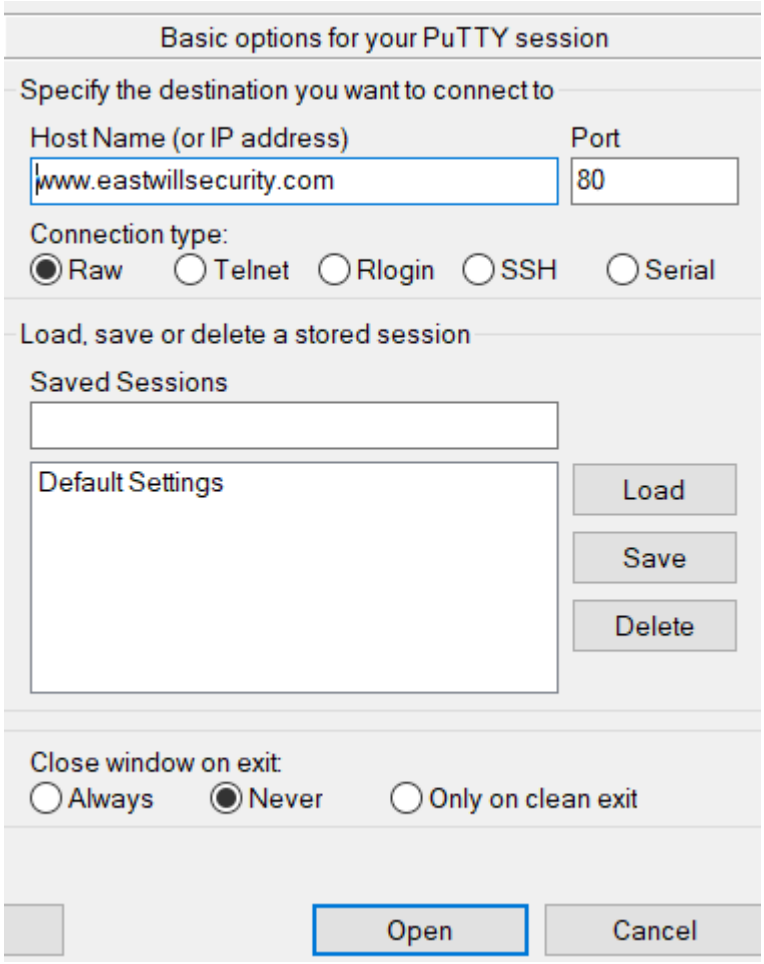
(Skjermdump av nettleseren)

3. Stoppet opptaket av nettverkstrafikken
4. Pinget i www.eastwillsecurity i kommandolinjen for å finne IP-adressen
5. La inn filter i Wireshark: «**ip.addr==77.111.240.75 && http**»
6. Valgte inngående http (response fra serveren) og tok skjermdump av http-headeren og datainnhold:



3d)

Logget inn:



La inn følgende:

POST /path/tk1104/oppgave3d.php HTTP/1.1

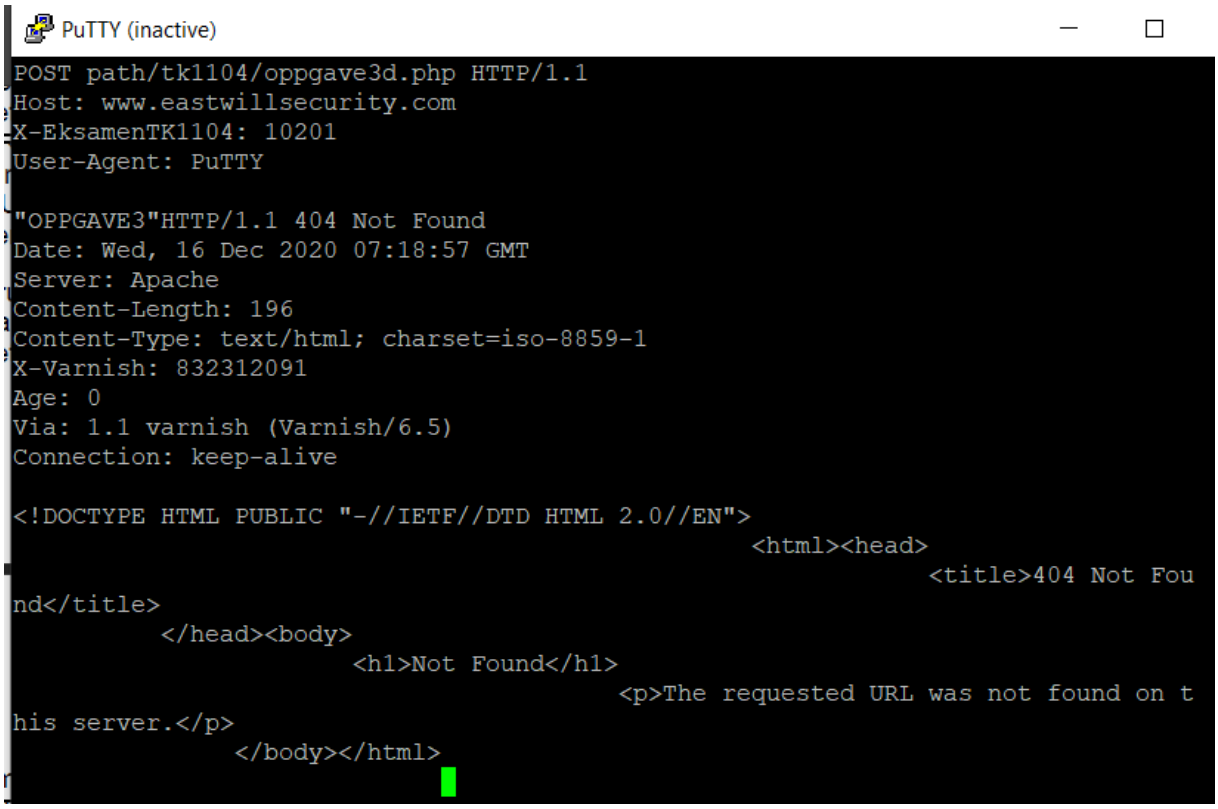
Host: www.eastwillsecurity.com

X-EksamenTK1104: 10201

User-Agent: PuTTY

“OPPGAVE3”

Resultat:



4a) ARPA Net

Under den kalde krigen var det spenninger mellom USA og Sovjetunionen. I 1957 slo Sovjetunionen USA i romfartskappløpet da de skjøt opp verdens første satellitt, Sputnik, opp i rommet.

For å sikre teknologisk overlegenhet opprettet USA i 1958 forskningsbyrået Advanced Research Projects Agency (ARPA), underlagt forsvarsdepartementet. ARPA ønsket å effektivisere formidlingen og samordningen av forskningsarbeid mellom forskerne i USA.

I 1962 oppdaget USA at Sovjetunionen hadde utplassert langdistansemissiliser på Cuba som kunne nå USA. Det oppstod reell frykt for atomangrep på amerikansk jord. Militæret ønsket derfor en pålitelig måte å kommunisere på ved et eventuelt atomangrep. De trengte en desentralisert nettverksarkitektur som ikke blir slått ut hvis en nettverksnode blir slått ut.

I 1966 begynte ARPA, sammen med næringslivet, å utvikle et datanettverk som kunne svare på behovet til forskerne og militæret. i 1969 ble ARPA Net lansert, utviklet av BBN Technologies (Bolt, Beranek and Newman).

Det var flere fagmiljøer i Europa som også jobbet med å utvikle egne datanettverk. Etter hvert begynte de ulike datanettverkene å koble seg sammen (nettverket av nettverk), og i dag alle koblet sammen i samme datanettverk.

Det har påvirket verden enormt. Vi har aldri hatt tilgang til så mye informasjon. Dette har gjort at teknologiutviklingen akselsert på alle fronter.

4b)

Src port = 2114	Dest port = 53
Size = 12	Checksum = ???? ???? ???? ????
Data (binært) = 1100 1111 1111 0101 1100 1111 0001 0000	

1: regne om desimaltallene til binær i 16-bit:

Src port = 0000 1000 0100 0010	Dest port = 0000 0000 0011 0101
Size = 0000 0000 0000 1100	checksum = ???? ???? ???? ????
Data = 1100 1111 1111 0101	1100 1111 0001 0000

2: Summerer sammen alle 16-bit delene

	0000 1000 0100 0010
+	0000 0000 0011 0101
=	0000 1000 0111 0111

	0000 1000 0111 0111
+	0000 0000 0000 1100
=	0000 1000 1000 0011

	0000 1000 1000 0011
+	1100 1111 1111 0101
=	1101 1000 0111 1000

	1101 1000 0111 1000
+	1100 1111 0001 0000
=	1 1010 0111 1000 1000

3. Summeres inn overflowen:

	1010 0111 1000 1000
+	0000 0000 0000 0001
=	1010 0111 1000 1001

4. Tar enekomplementet (flipp) av summen

	1010 0111 1000 1001
Checksum =	**0101 1000 0111 0110**

4c)

1. **Omregning til binær**

IP-adresse 1 omregnes til binær

10	112	42	29
0000 1010	0111 0000	0010 1010	0001 1101

Subnet mask omregnes til binær

255	255	248	0
1111 1111	1111 1111	1111 1000	0000 0000

IP-adresse 2 omregnes til binær

10	21	47	2
0000 1010	0001 0101	0010 1111	0000 0010

2. **Finne nettverksadressen til IP-adressene ved å ta AND operasjon med subnet mask**

IP-adresse 1 & subnet mask

	0000 1010	0111 0000	0010 1010	0001 1101
&	1111 1111	1111 1111	1111 1000	0000 0000
=	**0000 1010**	**0111 0000**	**0010 1000**	**0000 0000**

IP-adresse 2 & subnet mask

	0000 1010	0001 0101	0010 1111	0000 0010
&	1111 1111	1111 1111	1111 1000	0000 0000
=	**0000 1010**	**0001 0101**	**0010 1000**	**0000 0000**

Vi ser at IP-adressene ikke har samme nettverksadresse, og det betyr at de ikke er på samme nettverk.

De to datamaskinene kan ikke sende data til hverandre, da de ikke er på samme nettverk.

4d) Hva er NAT?

NAT står for Network Address Translation. Det er en innbygd funksjon i routeren for oversette offentlig IP-adresse til lokal IP-adresse og mottsatt.

Dette var løsningen på problemet med at det er for få unike IP-adresser med IPv4. IPv4 gir ca. 4,3 milliarder unike IP-adresser, men dette er ikke nok til hver maskin/device. Det betyr at flere maskiner kan dele samme offentlig IP-adresse

Derfor har hver maskin en lokal IP-adresse, som brukes internt i det lokale nettverket, og en offentlig IP-adresse som maskinen deler med andre.