Noter til eksamen 2

Alle spørgsmål 2

Generelt 2

1 Computer Architecture 3

M1. Digital logic circuits 3

M2. Data and program representations 4

M3. Processor architecture 5

M4. Virtual memory 7

M5. Caches 8

C1. Binary numbers 9

C2. Architecture 9

C3. Microcode 9

C4. Instruction sets 10

C5. Assembly languages 10

C6. Memory 11

C7. Size units 11

C8. Technology 11

C9. Memory 12

2 Computer Networks 13

M1. The Internet 13

M2. Application Layer 15

M3. Transport Layer 17

M4. Explain TCP, ﬂow control, and congestion control 19

M5. The Network Layer 20

M6. Routing Algorithms 21

M7. The Link Layer and LANs 22

M8. MAC layer protocols 23

C1.Application layer 25

C2. TCP/UDP 25

C3. Routing 25

C4. DHCP 26

C5. MAC layer 26

C6. Wireless LANs 26

# Noter til eksamen

## Alle spørgsmål

Mx: Main (primary) questions.

Cx: Control (secondary) questions.

## Generelt

[www.Smartlearn.com](http://www.Smartlearn.com) !!!

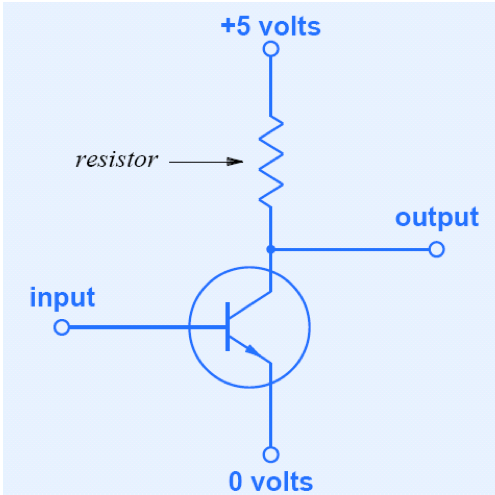
### 1 Computer Architecture

#### M1. Digital logic circuits

How are digital logic circuits made and how are they used to build boolean

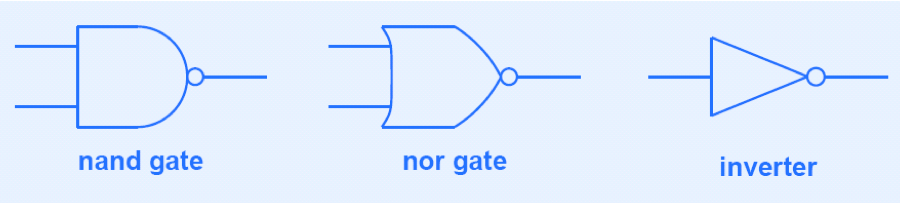
functions and higher level functions? (Keywords: transistor, dissipation, logic gates, truth tables.)

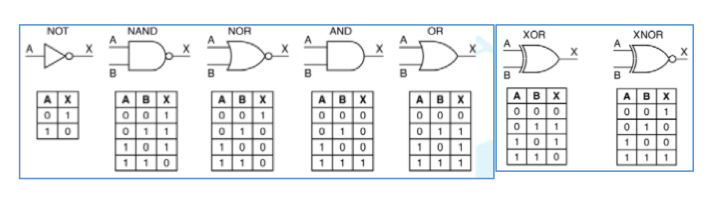
***Transistor:*** Grundelementet i en chip er ***transistorer***. Disse fungerer altid med ***to*** linier af lav strøm f.eks. 0 og 5 Volt. De fungerer som kontakter, der outputter 0 eller 1.



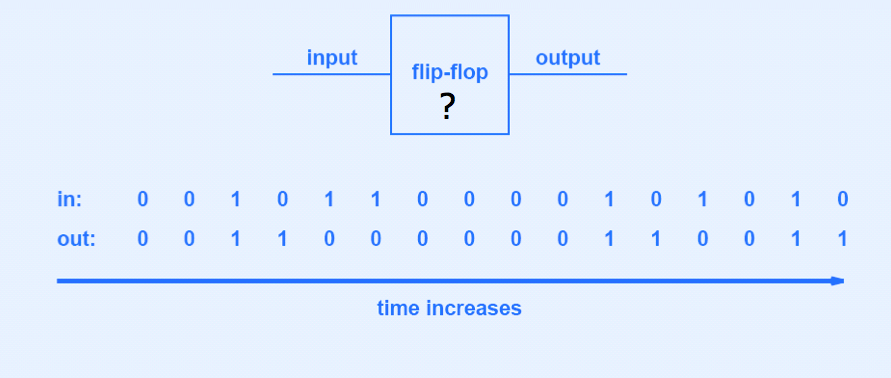
***Logic gate + truth table:*** Med transistorer kan laves logiske ***gates***. Eksempler på gates kan være ***NOT, AND*** eller ***OR*** (eller ***NAND, NOR).*** Disse kan være forbundet med hinanden f.eks. i serier. (TTL) (f.eks. VLSI Very Large Scale Integration) = ***Boolske udtryk!***

Ved at forbinde flere gates kan man skabe boolske udtryk, hvor outputtet af hver gate er inputtet i næste.

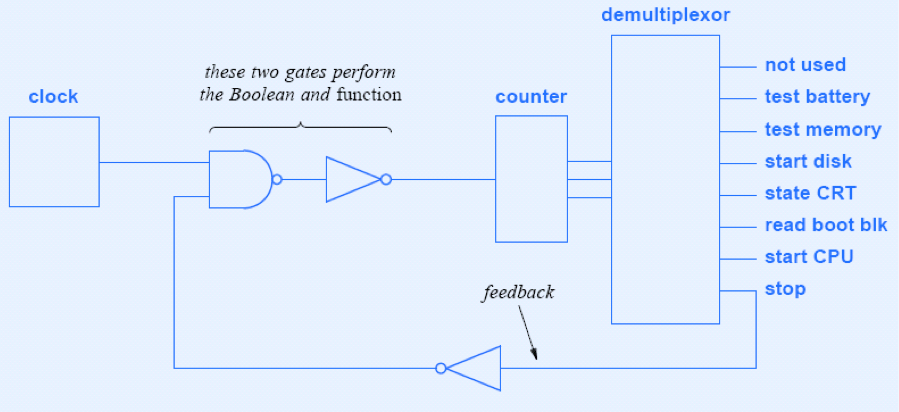




For at ***”bibeholde”*** udtryk over tid (hukommelse), så kan der anvendes et kredsløb som f.eks. et flip-flop kredsløb.



***Clock:***



#### M2. Data and program representations

How are data and programs represented? Keywords: binary numbers, opcode,

operands.

Binary numbers:

En computer opererer med binære tal (0, 1), der kan relateres til boolske værdier (Sand, Falsk).

Alle tegn og tal omregnes i computeren til binære tal, hvor laveste værdi er én bit.

En samling af 8 bits er en byte. En byte er det man for det meste bruger til at repræsentere alfabetiske, numeriske og alfanumeriske data.

Data er en kombination af karakterer, tal og symboler, disse er inddelt i tre forskellige typer:

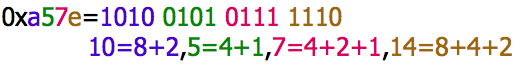
***alfabetisk data, nummerisk data, og alfanumerisk data***.

***Alfabetisk data*** er brugt til at repræsentere alfabetet. Den indeholder de store bogstaver A-Z, små bogstaver a-z og mellemrum. Alfabetisk data er også kendt som ikke numerisk data.

***Numerisk data*** indeholder 10 cifre 0,1,2,3,4,5,6,7,8,9 two symboler + og – og et decimalt symbol. Der er forskellige typer af tal systemer der bruges til at repræsentere numerisk data.

***Alfanumerisk data*** er blanding (bogstaver og tal).

Disse numeriske systemer er decimal tal system, binære tal system eller f.eks. hexadecimal tal system.



Binære tal kan være signed integers (kan være negative) eller som unsigned integers (kun positive). Ved signed er det den højeste bit, der koder tegnet og bestemmer, om tallet er positivt eller negativt

(“1” = “-“).

Eksempel på heltal, der skal kodes om kan være tallet 37 = 2(5)+2(2)+2(0) = 100101

***Operand & Opcode:*** Opcode er en integreret del af f.eks. en CPU, der specificerer handlinger, der kan udføres. Det kan f.eks. være matematiske beregninger som ***”+”***.

En ***operand*** er data/input, der er med et stykke opcode.

Hvis to tal som 7 og 3 skal lægges sammen, så gøres det vha. opcode for ”+” sammen med ***Operands*** ”3” og ”7”.

**ASCII = 128 characters (7bit) 2^7**

**Unicode = 65536 characters (16bit) – 2^16**

***Overflow & underflow:***

***Overflow*** sker når resultatet bliver større end det antal af bits der var tildelt.

***Underflow*** er, når der læses hurtigere fra en buffer, end der bliver skrevet til den. (f.eks. når en buffer forbinder to elementer som to drev)

#### M3. Processor architecture

What is the general structure of processors and how do they work? Keywords:

execution units, fetch-and-execute cycle, pipeline.

***Processorens opbygning og funktion:***

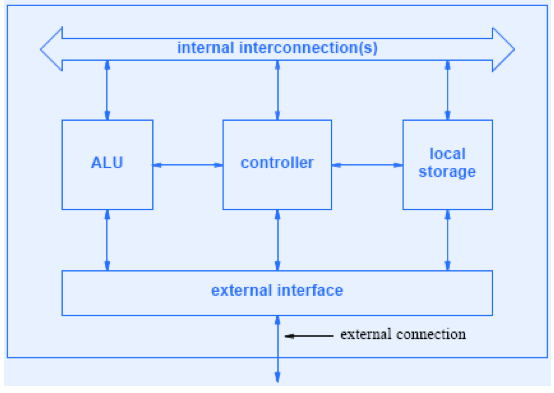
***Controller,***

***Beregnere (f.eks. ALU),***

***Lagre (registre eller cache),***

***Busser***

***interface (I/O).***



Complex Instruction Set

Reduced Instruction Set

Er for det meste bygget op som enten RISC eller CISC. I dag er det ***en blanding*** af de to.

Regiters er en høj hastigheds opbevarings enhed, i processoren. Der er få af dem og de har en begrænset størrelse med **basis** **operationerne** **fetch and store**.

***Execution units:***

**?????????**

***Fetch-execute cycle:***

Denne cyclus bliver brugt af en computers ***CPU*** til at gennemkøre et program fra start til slut og udføre hvert enkelt step på vejen.

Repeat forever {

***Fetch***: next instruction

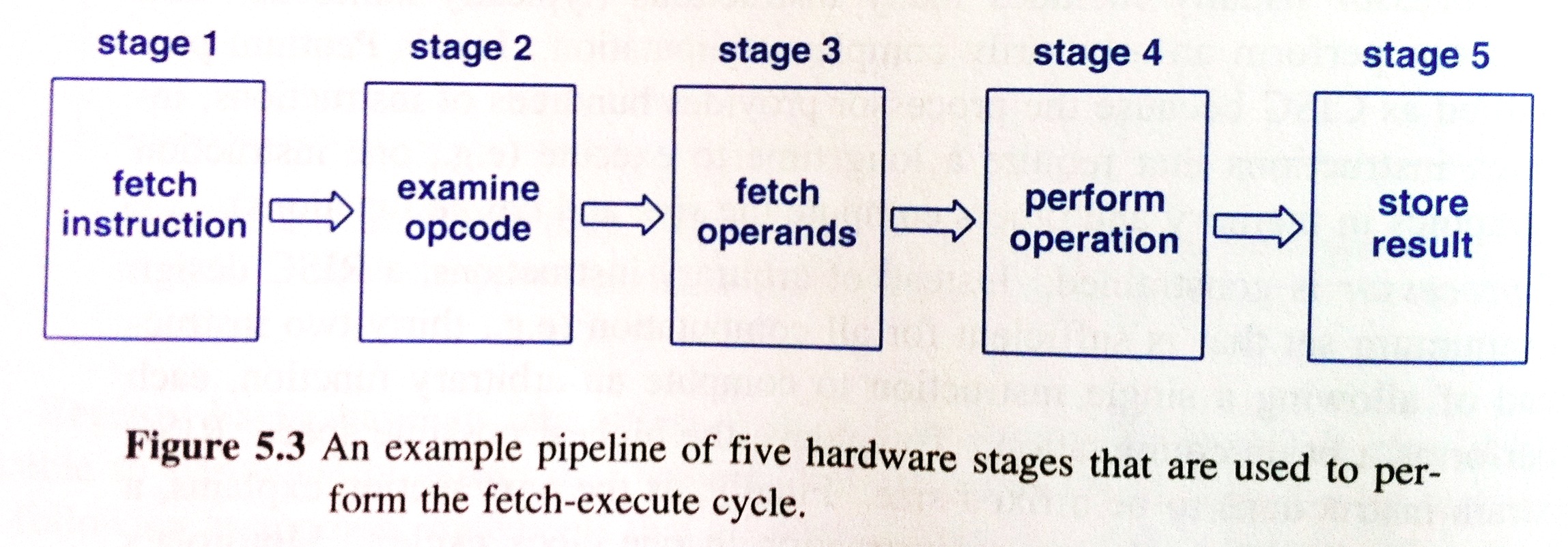
***Execute***: perform the step of the program.

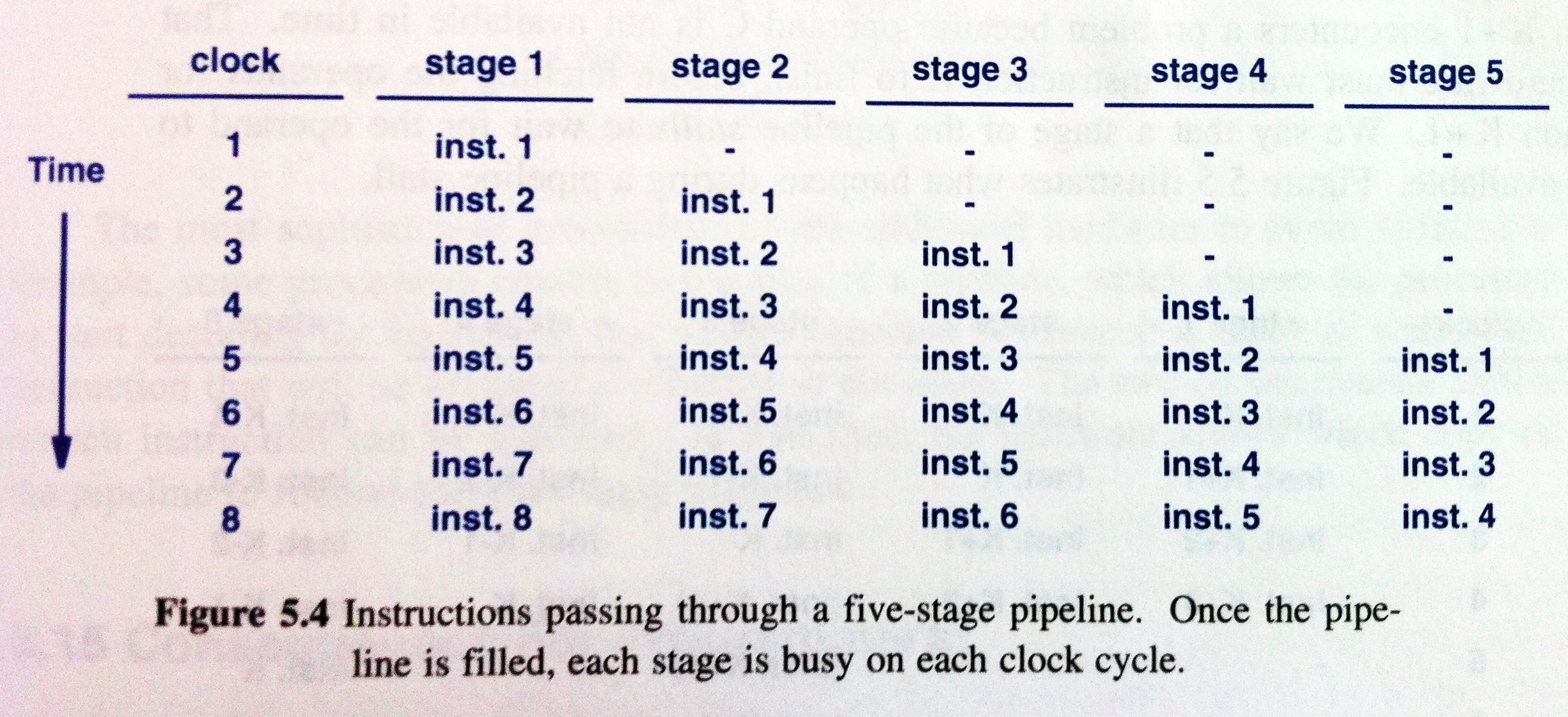
}

***Pipeline:***

En pipeline er en teknik der bliver brugt I design af computer og andre digitale sammenhæng for at øge det antal af instruktioner der kan udføres på samme tid.

Ideen i et pipeline system er at dele hver enkel aktion op i flere dele, hvilket er med til at gøre maskinen hurtigere da den kan lave flere ting på en gang. Et eksempel på dette kan fx være samlebåndssystemet, hvor man laver lidt af gangen hele tiden istedet for kun at fokusere på en ting af gangen.





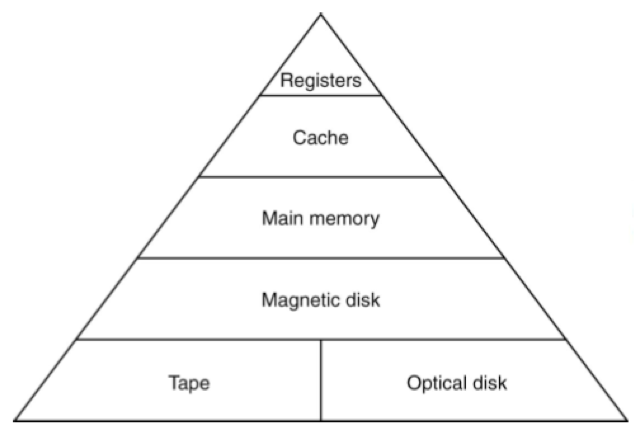
#### M4. Virtual memory

Explain the virtual memory system and its hardware support. Keywords: MMU,

execution modes, memory protection, paging, TLB.

Der findes forskellige former for hukommelse i f.eks. en almindelig computer:

***RAM, ROM, Flygtig, hurtige/langsomme.***



***MMU:***

Memory Management Unit er en komponent, der administrerer den virtuelle hukommelse. Den er bindeledet mellem den virtuelle og den fysiske hukommelse.

Fordelen ved en MMU er:

* Nemmere for programmører, da der kun er én samlet hukommelse
* Kombination af f.eks. flere RAM blokke til én hukommelse.
* mindre fragmenteret hukommelse ***(”spildt”)***
* større sikkerhed (paging pg segmentering)

***Execution modes:***

???

***Paging:***

Den virtuelle hukommelse deles ind i ***pages***.

***Memory protection:***

***Paging og segmentering*** af hukommelsen er en måde at beskytte den på. Ved at inddele i f.eks. 4 segmenter, beskyttes systemet, hvis et program udfører en fejl i ét af segmenterne.

Ligeledes beskyttes ved at inddele den virtuelle hukommelse i ***pages***, som MMU’en gør.

***TLB:***

Translation Lookaside Buffer er en buffer (***cache hukommelse***), der holder et vist antal oversættelser. Hvis informationen ikke er tilgændelig i TLB hentes den derefter i ***pages*** i hukommelsen. (langsommere!)

#### M5. Caches

Explain the role of caches, how (and why) they work, and their impacts for

programmers. Keywords: locality, write-through and write-back caches, associativity, LRU policy, memory mountain experiment.

Locality:

“***Temporal locality”*** er når data og/eller ressourcer bliver ***genbrugt / efterspurgt indenfor korte intervaller*** med henblik på at optimere hastighed (som f.eks. i cache hukommelsen)

***Locality! = Placering tæt på CPU’en***

Write-through and write-back caches:

**Write-through**: CPU skriver **først** til cachen og **cachen skriver så** i RAM blokken.

**Write-back**: CPU skriver **kun** til cachen, men skriver kun til RAM blokkene, **hvis der opstår plads problemer** i Cachen. Lager opdateres senere når cache opdateres.

LRU policy:

**Least Resently Used** metoden gør at det der bliver brugt **mindst bliver slettet først** og anvendt til nye data. Her **tæller operativsystemet** over hvor mange gange en page bruges **indenfor et tidsinterval**. Her kan der udtrækkes statistikker over cachingen. Ved Not Recently Used tælles ikke, men funktionen er ellers den samme.

Dette er et alternativ til ”gammeldags” ***FIFO*** metode til styring af, hvilken data, der skulle erstattes.

Associativity:

Et alternativ til direct cache, hvor den samme data gemmes to eller flere steder. Dette er et trade off. Det kræver ekstra ressourcer at gøre, men det minimerer conflict misses (hvor data er erstattet af nye i cachen).

Memory mountain experiment:

????

#### C1. Binary numbers

How are integers encoded?

Heltal kodes som binære tal. Det kan enten være som signed integers (kan være negative) eller som unsigned integers (kun positive). Ved signed er det den højeste bit, der koder tegnet og bestemmer, om tallet er positivt eller negativt (“1” = “-“).

Eksempel på heltal, der skal kodes om kan være tallet 37 = 2(5)+2(2)+2(0) = 100101.

#### C2. Architecture

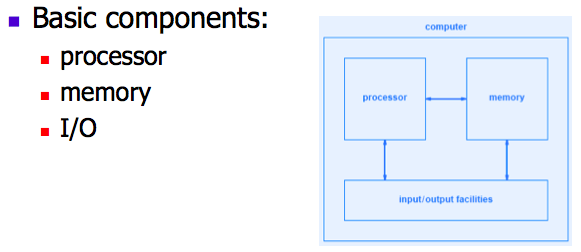
What is the Von Neumann architecture and what is its bottleneck?

Hans desgin går på fire dele**: ALU (A**rithmatic**L**ogic**U**nit)**, Control Unit, Memory and Input/output**.

**Von Neumanns** ide bygger kun på **én memory** (og bus) for **både data lagring og program lagring**.

Dette giver problemer da man **ikke kan hente instruktioner og data på samme tid**. En **løsning** for dette, er at tilføje en hurtigt **cache hukommelse**.

Grundlæggende koncept - programstyret. (John Von Neumann - matematiker). Modellen passer til de fleste moderne processorer. Grundlæggende komponenter: processor, hukommelse, I / O.



#### C3. Microcode

What are microcodes and macrocodes and why do we have both?

**Microcode** er det **interne sprog I CPUen**. **Macrocode** er det **synlige maskinkode** (Programmers reference), som kan ses udad til. I CPUen er der en **microcontroller** som **afkoder** og **udføre** **macro** **instruktionen**.

#### C4. Instruction sets

Compare the CISC and RISC instruction sets.

***CISC – Complex Instruction Set Computing***:.

Denne har mange måder at adressere på og mange operationer. Instruktionerne skal kunne fortolkes.

Komplekse instruktioner kan blive nedbrudt til mindre dele og blive udført som en sekvens af mindre instruktioner.

***RISC – Reduced Instruction Set Computing:***

Dette sæt er færre instruktioner, som ikke behøver at blive fortolket. Dette giver en hurtigere udførelses tid pr. instruktion, men kræver flere instruktioner i forhold til CISC.

I dag er mange CPUer hybride på den måde at komplekse instruktioner er fortolket som i CISC og simple instruktioner er udført direkte.

Fordel ved RISC er at hardware kan optimeres meget pga. det begrænsede instruktionssæt, men kan så ved afvikling koste ekstra ved at de samme instruktioner skal udføres mange gange for at få det samme resultat, som ved en CISC.

#### C5. Assembly languages

Which types of instructions are available? Give an example of common programming pattern (pseudo-assembly code).

Assembly language er et ***lav niveau sprog*** der er meget tæt på maskinkode. Alternativt findes f.eks. høj niveau programmerings sprog som C# eller Java.

Hver erklæring producerer ***præcis*** én maskine instruktion.

Assembly sprog er hardware afhængig og OS orienteret. Hvert assembly sproget er tilknyttet en processor type.

***Eksempler er:***

ADD, SUB, MUL og DIV

***Eksempel på pseudo kode:***

***If*** student's grade is greater than or equal to 2

Print "passed"

***else***

Print "failed"

#### C6. Memory

What is data alignment and why is it important?

Hvis data ikke tilpasses (***misaligned***), så skal CPU’en arbejde ekstra for at få adgang til dataen, dette sker typisk ved at den skal loade to dele data, fjerne de bytes der ikke har interesse og forbinde dem.

Denne process sløver performancen ned og spilder CPU’en kraft bare for at få det rigtige information fra hukommelsen.

Derfor er det vigtigt som programmør at gemme informationen ordentlig, dette kan også betyde at vi bliver nødt til at indsætte tomme felter hvis en data fx kun fylder 3 byte for at undgå misalignment.

#### C7. Size units

Why is it important to use powers of 2?

Pga. det **binære talsystem**, **0 og 1**. Da der kun er **to muligheder** med binære tal, opløfter man det **antal af tal** pladser **ønsket *n* med to**. På denne måde kan man udregne hvor mange muligheder man har ved et vist antal *n*. Fx fire pladser regnes således: **24 = 16**. Altså er der **16 tal kombinationer** ved brug af **4 pladser**.

Et spørgsmål fra censor kunne være: kunne man så ikke anvende f.ske. 10-tals-systemer som vi mennesker bedre kan forstå og så få 10 gange så meget performance?

Svaret: nej for en elektrisk component som en **transistor** kan **kun** **have** **2 værdier**, **strøm/ikke-strøm**, repræsenteret ved 0 og 1, derfor er det binære system begrænsningen.

#### C8. Technology

Compare SRAM and DRAM technologies.

RAM er en fysisk lagerenhed hvor computeren lagrer de data den arbejder med her og nu.

***DRAM – Dynamic***

Lagrer hver bit i sin egen capacitor (transistor) og denne lagring skal have strøm for at kunne opretholde informationen.

Bruges i bærbare, stationære, konsoller og lignende…

***SRAM – Static***

Er statisk ram og behøver derfor ikke en jævn tilgang af strøm for at opretholder information. Hver bit er repræsenteret i 4 transistorer, der er koblet i par.

Bruges i form af f.eks. hukommelseskort til kamera, usb-lagerenheder og lignende.

#### C9. Memory

Explain the term memory hierarchy.

**Computerens hukommelse** er inddelt i **5 levels,** etsåkaldt **hieraki.**

* **CPU-registers** (cpu full speed).
* **Cache** (32kb to few megabytes).
* **Main memory** (from 16mb to 10 gigabytes).
* **Magnetic disk**.
* **Tapes and optical disks**.

Dette hieraki er den traditionelle måde at gemme data på. Den er baseret på tre vigtige faktorer:

1. Jo længere vi kommer ned i hierarkiet, **des længere tid tager det for CPUen at tilgå dataen.**
2. Man får **mere hukommelses plads**, jo længere ned man kommer i hierarkiet.
3. Man får **mere plads for pengene**, jo længere ned man kommer i hierarkiet.

### 2 Computer Networks

#### M1. The Internet

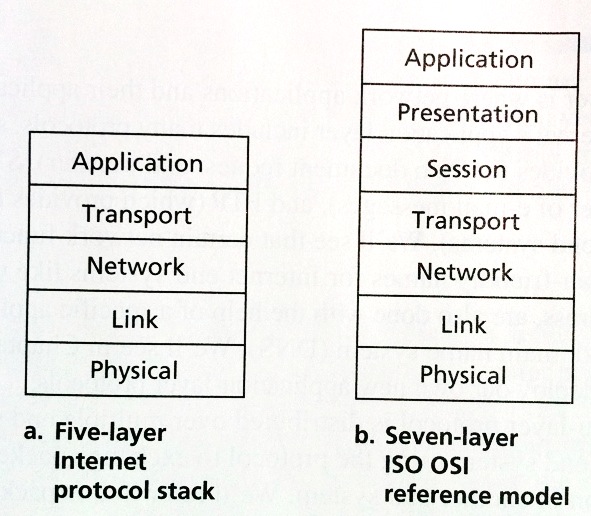
Explain the OSI model, how is it related to the Internet Protocol Stack (5 layers). Explain Circuit and Packet switching (pros and cons, how do they diﬀer?).

Internet Protocol Stack (5 layers):

Internet Protocol Stack er opbygget af 5 layers, hver enkelt protokol hører til et layer. Hvert layer håndterer specifikke opgaver, og komunnikerer med laget før.

OSI modellen modellen (Open Systems Interconnection) er en videreudbyggelse af denne hvor der er blevet tilføjet yderligere to lag:

***Presentation layer*** og ***session layer.***

******

**Application Layer: Function: network process to application.**

***HTTP*** (web dokumentation forespørgsel og overførsel)

***SMTP*** (overførsel af e-mail beskeder)

***FTP*** (overførsel af filer imellem to systemer)

***DNS*** (domain name service - En DNS-server kan oversætte mellem IP-adresser og domænenavne for et begrænset antal domæner eller den kan kontakte andre servere for at få oversat en vilkårlig adresse)

Bruges til at udveksle pakker af informationer mellem to end systemer med samme protokol. Disse pakker af informationer er kaldet ”***messages***”.

***Transport Layer:* Data repræsentation …….. (**Virker Process-to-Process!)

***TCP* (Transport Control)**

Forbindelsesservice til applikationer med **flow control og congestion control**

Sikrer at alle pakker kommer frem, og at intet går tabt.

Anvendes til [**World Wide Web**](http://en.wikipedia.org/wiki/World_Wide_Web)**,**[**email**](http://en.wikipedia.org/wiki/Email)**,**[**remote administration**](http://en.wikipedia.org/wiki/Remote_administration)**og**[**fil**](http://en.wikipedia.org/wiki/File_transfer) **overførsel**.

***UDP (User Datagram****)*

Giver ingen garanti for at dataen kommer frem – eller afsenderen får ikke besked hvis data ikke kommer frem.

UDP bruges eksempelvis på **internet radio, VoIP, IPTV** hvor hastigheden er vigtigere end at alle pakker kommer frem

**Network Layer (IP layer): The postman …… (**Virker Host-to-Host!)

Dette lag er ansvarlig for at flytte ***”Datagrams”*** fra en host til en anden host. Transport layer protokollen sender et transport-layer segment, og en destinations adresse til netværks layer. Netværks layer sørger derefter for at levere dette segment til transport layer i destinationens host.

Dette bliver typisk gjort ved hjælp af en ***IP adresse***. En IP adresse er en unique adresse som hver internet bruger har og kan derfor bruges til at spore folk.

***Datagram: Header (IP + routning protokol)***

Internettet har mange routing protokoller, fordi at internettet er et netværk af netværk. Hvis man fx skal sende et segment fra et netværk til en person på et andet netværk – så vil man først sende det til netværket og derefter ud til modtageren.

**Link Layer:**

***Link layeret står for at sende pakken til den næste node på routen***.

Internet netværk layer sendte som beskrevet før et datagram igennem en serie af routere imellem kilde og destinations stedet. For at flytte en packet fra en node til den næste node i routen, må netværks layer stole på servicen i link layeret. Link layeret er mere fokuseret på at arbejde med 2 eller flere nodes af gangen.

**Physical Layer:**

Når link layerets job er at få pakken fra et netværk til et andet netværk, så er det physical layerets job at få ***de individuelle bits*** fra en node til den næste. Physical layer ser ikke på det store hele, men kun at pakken kommer fra node A til node B.

**OSI modellen:**

I slutningen af 1970erne blev OSI – Open systems interconnection oprettet. Forskellen mellem OSI og IPS modellen er at OSI har to layer mere:

**Presentation layer**

Dette lag fortolker den data der skal sendes, ved at ***sammenligne, kryptere og beskrive dataen***. På denne måde skal de forskellige applikationer ikke forholde sig til hvilket data format de modtager.

***Session layer***

Dette lag indeholder afgrænsning og synkronisering af data udveksling. Dette betyder at ***session laget*** holdet styr på hvis tur det er til at sende og om forbindelse er åben eller ej. Hvis forbindelsen går ud, så vil det også være sessionslaget, der prøver at genoprette det.

Circuit and packet switching:

**Circuit switching (FORBINDELSE)** fungere på den måde at man skaber en forbindelse mellem to enheder før man sender beskeden. Ved denne metode skaber man en dedikeret forbindelse, uden forstyrrelse, hvilket også fjerner mest forsinkelse.

Desværre tager det tid at skabe forbindelsen og der kan kun være én ad gangen.

***Et eksempel*** på dette kan være et telefon system, hvor man indtaster et nummer også sender centralen en videre til det nummer.

**Packet switching (BARE SENDT)**: Man deler dataen op i flere dele også kaldet packets som så bliver sendt over netværket. Hver gang en package kommer til en router, kigger routeren på den og sender packets videre i den rigtige retning.

***Fordelen*** ved dette er at man udnytter trafikken meget bedre.

***Ulempen*** ved dette er dog at der er flere chancer for at data går tabt undervejs, hvilket vil lede til at man bliver nødt til at sende det igen.

#### M2. Application Layer

In the Internet Protocol Stack, What does the Application Layer do? Explain

any two of the following protocols, HTTP, FTP, SMTP/POP3/IMAP, and DNS.

Application layer in Internet Protocol Stack:

Application Layer: HTTP/SMTP/FTP/DNS protokoller. Applikation lagets protokoller bruges til at udveksle pakker af informationer imellem to end systemer med samme protokol. Disse pakker af informationer er kaldet ”Messages”.

***HTTP (The Hyper Text Transfer Protocol)***:

Denne protokol bruges til at hente websites til klienten fra serveren.

Det fungere således at når en klient anmoder om at se et website, så sender HTTP en anmodning til serveren. Serveren svarer ved at sende det ønskede objekt retur. ***HTTP bruger TCP***, som skaber pålidelige data overførelser til HTTP.

Dette sikrer at alle forespørgsler sendt fra klienten til hosten er modtaget intakt og visa versa.

(HTTP er en stateless protokol fordi at ***serveren ikke husker*** information omkring klienten. Dette betyder også at hvis en klient requester den samme fil to gange i træk, så ville serveren sende den begge gange.)

***Persistent HTTP:*** Når man bruger en persitstent forbindelse, så bruger man den ***samme TCP*** forbindelse for hver enkel forespørgelse/svar – ***dette er hvad man bruger som standard***.

***Nonpersistent HTTP:*** Når man bruger en ikke persistent forbindelse, så ***opretter man en ny TCP*** forbindelse hver gang man forespørger/svarer – ***dette kan en webside konfigureres til at gøre***.

Nonpersistent forbindelser kan være langsommere end persistent. RTT er den tid det tager for en pakke at flytte sig fra klienten, til hosten og tilbage til klienten igen. Når man bruger en ikke persistent forbindelse, så skal hvert enkelt object på en webside bruge 2 RTT, og hvis man bare en persistent så skal man kun bruge en.

***RTT (round-trip-time)***:

Den tid der går fra klienten requester til klienten modtager serverens svar.

***FTP (file transfer protocol):***

Denne protokol bruges til at overføre filer imellem en bruger og en fjern host. Brugeren skal identificeres af serveren med et brugernavn og password.

FTP bruger ***to TCP forbindelser***. Den første bruges til at identificere brugeren og den anden til at overføre de aktuelle filer.

Data forbindelse er en ***nonpersistent forbindelse***.

SMTP/POP/IMAP:

SMTP = Simple Mail Transfer Protocol

Ingen login. Bruges ofte til spam! 7 bit protocol (ASCII). Send protocol.

POP = Post Office Protocol

Login. Hentes ned dog slettes fra server. Bruger TCP. Hent protocol.

IMAP = Internet Message Access Protocol

Bliver på serveren med mindre du vælger at slette. Mapper er muligt.

DNS:

Domain Name Server. Oversætter IP til navne…

#### M3. Transport Layer

What is the main function of the transport layer protocols? How does TCP

implement the reliable transmission in the Internet?

Main function of transport layer protocols:

Den generelle funktion I transport lageret er at skabe en forbindelsesservice mellem en klient og en server.

***TCP (Transmission Control Protocol)***

Ideen i TCP er at den lægger fokus på god service når det kommer til forbindelse, dette betyder at den garanterer levering af applications lagerets beskeder til destinationen.

* ***Ordnet levering*** – pakkerne ankommer til modtageren i samme rækkefølge som de blev afsendt.
* ***Flow control*** (sender og modtagers ***hastighed matcher*** – dette kan være et problem hvis afsender computeren sender hurtigere end modtager computeren kan opfange dataen.)
* ***Congestion control*** (brugt til at kontrollere data flowet når forbindelsen bliver overbelastet) Dette kan bl.a. gøres ved at dele pakkerne op i mindre segmenter.

Bruges bl.a. til internet, email og filoverførsler

**UDP (User Datagram Protocol)**

UDP giver en forbindelsesløs service til applikationer. Det er en no-frill service der ***IKKE*** giver pålidelighed, flow control eller congestion control.

* Afsenderen får ikke besked om hvorvidt dataen er kommet frem.
* Transmissionshastigheden bestemmes af afsenderen uden hensyn til pakketab mv. Protokollen er dermed velegnet til overførsel af data, hvor ***konstant hastighed*** er vigtigere end fuldstændighed.

Bruges fx til internet radio

***Reliable data transfer (RDT):***

TCP har RDT ved brug af flow control, congestion control, sequence number, acknowledgement

***Sequence numre*** bliver brugt til at identificere hver enkel byte af data.

Bekræftelse bliver givet i form af ***feedback på hvorvidt alt dataen er blevet modtaget***. Et eksempel på dette kunne være at en fil på 10000 bytes bliver sendt ud i 10 forskellige TCP packets med 1000 bytes i hver. Den første pakke går tabt og derfor får modtageren kun 1000-9999. TCP vil her fortælle afsenderen at de første 1000 bytes mangler, så man bare kan sende dem istedet for at sende det hele afsted igen.

**Multiplexing and Demultiplexing:**

Multiplexing er en proces hvor adskillige analoge besked signaler eller digitale data streams er forbundet til et signal over et delt medium. Ideen i multiplexing er at dele en dyr resource.

Et eksempel på dette kunne være i telekommunikation, hvor mange telefonopkald kan blive forbundet via en ledning.

Multiplexing signalet er sendt ud over en kommunikations kanal, hvilket kan være et fysisk transmissions medium. Multiplexing deler kapaciteten af det lave level kommunikations kanal til højere level logisk kanaler. En for hver besked signal eller for hver data stream der skal overføres.

Den omvendte process kaldes Demultiplexing, hvilet bruges til at pakke de originale kanaler ud igen på modtager siden.

Dette vil altså sige at man kan ændre de langsomme signaler fra fx en computer til høj hastighedssignaler og derefter lave dem om igen når de kommer til modtager siden.

#### M4. Explain TCP, ﬂow control, and congestion control

Den generelle funktion I transport lageret er at skabe en forbindelsesservice mellem en klient og en server.

***TCP (Transmission Control Protocol)***

Ideen i TCP er at den lægger fokus på god service når det kommer til forbindelse, dette betyder at den garanterer levering af applications lagerets beskeder til destinationen.

* ***Ordnet levering*** – pakkerne ankommer til modtageren i samme rækkefølge som de blev afsendt.
* ***Flow control***

sender og modtagers hastighed matcher – dette kan være et problem hvis afsender computeren sender hurtigere end modtager computeren kan opfange dataen.)

* ***Congestion control***

brugt til at kontrollere data flowet når forbindelsen bliver overbelastet. Dette kan bla gøres ved at sætte sende hastigheden på pakkerne ned)

* ***Sequence numre*** bliver brugt til at identificere hver enkel byte af data.

Bekræftelse bliver givet i form af feedback på hvorvidt alt dataen er blevet modtaget. Et eksempel på dette kunne være at en fil på 10000 bytes bliver sendt ud i 10 forskellige TCP packets med 1000 bytes i hver. Den første pakke går tabt og derfor får modtageren kun 1000-9999. TCP vil her fortælle afsenderen at de første 1000 bytes mangler, så man bare kan sende dem istedet for at sende det hele afsted igen.

* Bruges bl.a. til internet, email og filoverførsler

I OSI-modellen er TCP det mellemliggende lag mellem Internetprotokol og applikationen.

Applikationer har som oftest brug for stabile datastrømme, hvilket Internetprotokollen ikke tilbyder. Den tilbyder blot levering af datapakker mellem to værter. I daglig tale benytter man dog i praksis forkortelsen ***TCP/IP*** som betegner sammenkoblingen mellem TCP og Internetprotokollen.

#### M5. The Network Layer

Explain VC/Datagram Networks, and any three of the following: IP, IP addressing, subnetmask, and NAT.

Virker Host-to-Host! .. “the postman”

VC Networks:

Virtual Circuit, VC, er sammenligneligt med et ***Circuit Switch netværk***, men hvor der anvendes virtuelle kredsløb. Pakkerne bevæger sig igennem nettet ad de same ruter og vil blive modtaget I same rækkefølge som de blev sendt. Der gives garanti for at modtageren er parat til modtagelse, da der skal modtages en kvittering for en opkaldspakke før der må sendes en datapakke.

***Datagram networks*:**

En grundlæggende overførsels enhed, som er forbundet med et ***packet-switched netværk***, hvor levering, ankomsttid og orden ikke er garanteret. ***Et datagram består af header og data områder***, hvor headeren indeholder den information der er nødvendig for at man kan route fra det computeren til destinationen, uden at være afhængig af forrige udvekslinger imellem computeren og netwærket.

Start adressen, slut adressen samt et type felt er at finde i headeren af et datagram. Ordet datagram bliver ofte betragtet som et synonym til packet, men der er nogle forskellige. Packet henvender sig til enhver besked formateret som en pakke, hvor begrebet datagram er reservered for pakker med upålidelig service. En upålidelig service giver ikke brugeren besked hvis leveringen fejler.

UDP bruges eksempelvis på **internet radio, VoIP, IPTV** hvor hastigheden er vigtigere end at alle pakker kommer frem

***IP – The internet protocol:***

IP er en protocol brugt ved kommunikation af data igennem ***packet-switched internet*** ved brug af TCP/IP. IP er den primære protokol i Internet layer i Internet Protocol Suite og har opgaven at aflevere protokol datagrams fra source host til destination host baseret på deres adresser.

***Ipv4*** er den dominerende protokol på nettet lige nu og bliver brugt til at definere adresse metoder og strukturer for datagram indkapsling.

***Ipv6*** er dog ved at komme mere frem.

***Subnet mask:***

Et TCP/IP netværk at opdeles Subnets. Standard subnet mask hedder 255.255.255.0 ligesom i IP-klasser har de en logisk opbyggelse. 255.0.0.0 (A net), 255.255.0.0 (B net) og 255.255.255.0 (C net). Her kan man se at A net er det første til i rækken og B net er det næste osv. Subnet mask giver mulighed for at have ”flere” netværk på et netværk, man kan betegne det som lag net. F.eks. en skole har et subnet til kontorerne (255.255.254.0) og eleverne (255.255.255.0), dette giver næsten uanede muligheder alene på et netværk, det eneste det kræver hvis man også har en Internet forbindelse er at routeren er klar over at der er flere subnets, for ellers vil det ikke modtage noget fra de subnets der ikke er indtastet i den. Hvis man har delt det op på denne måde gør det, det muligt at eleverne ikke har adgang til kontoernes computer og omvendt.

***NAT - Network Address Translation:***

NAT er en standard som gør det muligt at et lokalnetværk (LAN) bruger et sæt IP-adresser til intern trafik og et andet sæt IP-adresser til ekstern trafik. Ved at benytte alternerende sourceports kan man translatere et eller flere netværk til en offentlig IP-adresse. Den teknik kendes som many-one NAT, IP masquarading, PAT (Port Address Translation) og SUA (Single User Account).

#### M6. Routing Algorithms

Explain Link-state (Dijkstra’s), or Distance vector. Demonstrate Dijkstra’s (or distance vector). Explain how the two algorithms relate to RIP and BGP.

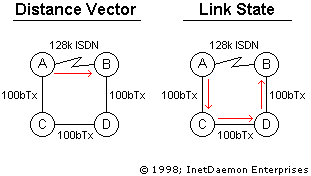
Distance vector og Link State er udtryk brugt til at beskrive routing protokoller, som er brugt af routere til at ***forwarde packets*** imellem netværk.

Ideen bag routing protokoller er at skabe finde den bedste og hurtigste måde at nå frem til et destinations netværk. Begrebet distance vektor og link state er brugt til fordele routing protokoller ind i to kategorier baseret på hvorvidt routing protokollen vælger den bedste routing sti baseret på distance eller interface. Eller vælger den bedste routing sti ved at kalkulere hvilken sti der er hurtigst (baseret på hastighed)

Link-state (Dijkstra’s):

Hvis man ser på figuren nedenunder. Hvis alle routere kørte med en distance vector protokol, så ville den vej der altid ville blive valgt være A – B igennem IDSN serial linket, også selvom at det link er 10 gange langsommere end den indirekte rute fra A,C,D,B.

En link-state protokol ville vælge ACDB ruten, fordi den bruger en hurtigere medium(100mb ethernet). I dette eksempel ville det være bedre at bruge en link state routing protokol, men hvis alle links i netværket er den samme hastighed, så ville distance vektoren være bedre.



Distance vector:

Distancen er som regel baseret på det antal hosts dataen skal igennem. Bruger metrics som hop count.

Vektor er hvor langt den skal. Baseret på trafik.

Distance vektorer bruger en distance kalkulation plus et netværk interface til at vælge den bedste vej til destinationsnetværket. Netværksprotokollen vil forwarde data efter den bedste vej er valgt.

Relation to RIP and BGP:

**RIP ( Routing information protocol):**

RIP er en dynamisk distance vektor routing protokol. RIP kalkulerer den bedste rute baseret på hop count. RIP kan ikke håndtere mere end 15 hop, dette er for at undgå routing loops.

**BGP (Border gateway protocol)**

BGP er en routingprotokol, der, takket være et intelligent design, har vist sig at skalere med Internettets udvikling. Er blevet de facto-standard for routing mellem autonome systemer.

Der eksisterer to overordnede former for BGP: ***E-BGP (external BGP)*** anvendes mellem autonome systemer, mens ***I-BGP (internal BGP)*** anvendes inden for grænserne af det enkelte autonome system (AS).  
  
BGP's store triumf som E-BGP på Internettet skyldes, at protokollen gør det muligt for AS'er at forbinde sig til og udveksle routingoplysninger med et eller flere andre AS'er uden hensyntagen til de store forskelle, der kan (og typisk vil) være mellem de enkelte AS'ers interne routingarkitekturer.

#### M7. The Link Layer and LANs

What is ARP protocol? Which information can a ARP table contain possibly?

Please describe in detail how the translation is done and how to send a datagram

to a host over the Internet.

ARP (***Address Resolution Protocol)***:

En protokol der bliver brugt til at ***oversætte IP-adresser (Network layer) til mac adresser (Link layer)*** og det er brugt når to hosts på samme netværk kommunikerer med hinanden. Første gang host A kontakter host B, sender host A en ARP forespørgsel i et broadcasting frame på netværket, hvor den efterspørger hvilken mac adresse der ejer IP adressen af host B.

Host B sender beskeden tilbage i en standard frame, og derefter vil host A så gemme adressen i ARP tabellen.

**ARP Table**

En ARP table indeholder informations, som viser IP-adresse, mac adresse, TTL (time-to-live) værdi. TTL værdien fortæller hvornår hver enkel mapping vil blive slettet fra tabellen.

How is translation done and how to send datagram to host:

**Translation**

ARP fungerer på den måde at den

* …først kigger på sin egen ARP tabel for at se ***om den givne IP adresse er at finde***.
* Hvis den ikke er på listen over de forbundne IP adresser, så forespørger ARP’en i et broadcasting frame hvor ***den efterspørger hvilken mac adresse der ejer IP adressen*** som man efterspørger.
* Derefter ***gemmer den IP adressen og mac adressen*** i tabellen.
* Hvis ARP’en ikke kan finde IP adressen, så returnerer den ***med en fejl meddelelse***.

**Send a datagram to a host over the internet**

Hvis man vil sende en pakke over internettet fra host A til host B:

* …så vil man først finde IP adressen på host B.
* Derefter bruger man ***DNS*** for at finde den router hvor host B’s IP adresse er på.
* Så sender man pakken til den router, og når routeren har modtaget it, så opretter den en ARP tabel og finder host B’s IP adresse.

#### M8. MAC layer protocols

We can classify multiple access protocol into three categories:

channel partitioning protocols, random access protocols and taking turns. Please explain their

principle respectively.

Channel partitioning protocols:

CPP indeholder ***tre*** forskellige typer af sendings metoder.

**TDM (Time-division multiplexing)**TDM opdeler tid i rammer og videre ind i tidsintervaller. Hver node er tildelt de enkelte tidsintervaller.  
Fordele er at det er fordelt på en fair måde.  
Ulemperne er at hele båndbredden ikke bliver brugt hvis alle tidsintervaller ikke sender pakker.

**FDM (Frequency-division multiplexing)**FDM opdeler dataen i forskellige frekvenser. Hver enkel node er tildelt en frekvens  
Fordele er at det er fordelt på en fair måde.  
Ulemperne er at hele båndbredden ikke bliver brugt hvis alle frekvenser ikke sender pakker.

**CDMA(Code-division mutliple access)**CDMA tildeler forskellige koder til hver enkel node. Hver node koder hver pakke med deres egen kode.   
Flere noder kan blive sedt på samme tid og den samme kanal, hvis modtageren kender afsenderens kode på pakkerne han modtager.

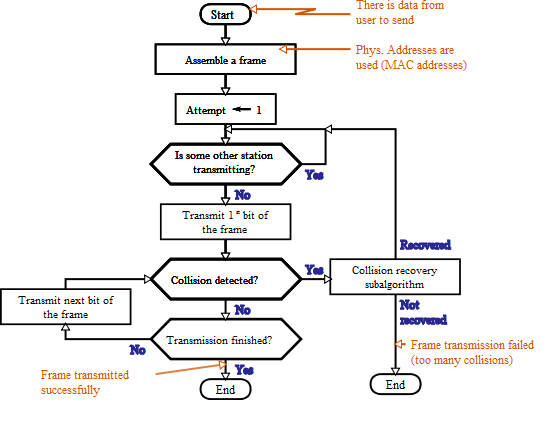
Random access protocols:

RAP indeholder ***tre*** forskellige typer af sendings metoder.

**Slotted ALOHA** fungerer på den måde at alle noder sender pakker i starten af tids intervallet. Hvis der er en kollision, så prøver de at sende pakkerne i det næste tids interval  
Fordele ved dette er at en node kan sende med fuld båndbredde, hvor ulemperne er synkroniseringen af tids intervallet.

**ALOHA** fungerer på næsten samme måde som Slotted ALOHA, forskellen er bare at på en kollision så venter den ikke på det næste tids interval, men på et specifikt tidsrum.  
Fordele ved dette er at den ikke behøver synkroniseringen, men ulemperne er at det kun er halvt så hurtigt som Slotted ALOHA.

**CSMA (casier sense multiple access)**

**

CSMA registerer om der er en anden node der sender pakker, og hvis dette er tilfældet så vil den vente et kort tidsinterval (backoff delay), før den prøver at sende pakken igen.   
Kollision registreres anvendes til at forbedre CSMA ydelsen ved at stoppe transmission øjeblikkelig, når en kollision detekteres og sandsynligheden for en kollision mindskes ved efterfølgende sendingsforsøg.

Taking turns:

TTP indeholder to forskellige typer af sendings metoder.

**Polling protocols**Skal bruge en master node, som kan putte de andre noder i en cirkel. En node, der er under den master node kan sende pakker. Master noden kan se når en node er færdig med at sende.  
Fordelene ved dette er at der ikke er nogen kollision og alt båndbredden bliver brugt. Ulemperne er at der kan være delay hvis master noden dør.

**Token passing protocols**En token kan skifte til forskellige noder, og den node der har token kan sende pakker.   
Fordele ved dette er at det er decentraliseret og effektiv. Ulempen er at en node kan ødelægge det hele ved ikke at sende token videre til andre nodes.

#### C1.Application layer

What is a socket? How can it be used?

Et applikations oprettet OS-kontrollerede interface (**en slags "dør"**), til hvilken applikationsprocessen kan både **sende og modtage beskeder** til / fra en anden applikationsproces. **Socket** er lokalt **identificeret** med et **portnummer**.

En socket er en slags **dør** imellem end **system programmet og transport lage**t. Før programmering er det vigtigt for udvikleren at bestemme hvilken port programmet skal bruge, **fx port 80 (standard for HTTP).**

#### C2. TCP/UDP

Isnt TCP always preferable to UDP since TCP provides a reliable data transfer

service and UDP does not?

**TCP (Transmission Control Protocol)** konstant **checker** for om alle **datapakker modtages** og derefter **justerer** **hastigheden** efter hvad der går godt, så kan man man sagtens komme ud for at køre med lavere hastighed end forventet og egentlig også end der er behov for, idet **routning** mellem **servere** på **internettet** konstant ændrer sig.

**Ved UDP** (User Datagram Protocol) er det **applikationerne**, der skal **checke** via den **checksum** protokollen leverer om **alle pakkerne er modtaget korrekt**.

UDP kan derfor være hurtigere end TCP til overførsel, og derfor kan det være godt at bruge i situationer hvor tab af enkelte pakker ikke betyder så meget (***f.eks. diverse streaming***)

#### C3. Routing

What is the diﬀerence between routing and forwarding?

**Forwarding:**

Forwarding af pakker, betyder at pakkerne bliver sendt i kopier til flere computere eller noder på netværket. Dette kan fx være broadcasting i form af internet radio eller internet broadcasting. Dette betyder også at det er modtageren (destinationen), som skal afslutte transmissionen medmindre at afsendings serveren går ned.

**Routing:**

Routing er mere målrettet. Ved routing bliver datagrammet på et tidspunkt afleveret fra source til destinationen som et resultat af gengivet forwarding. Her er det adresserne og netværkets opbygning der bestemmer hvor dataen skal sendes hen. Fx hvis man har en IP adresse der hedder 192.168.0.10, så ved man at computeren har host adressen 0.10 og at computeren befinder sig på netværket 192.168.0.0.

#### C4. DHCP

What is DHCP? How can it be used?

**Tildeler ipadresser, subnets og gateway adresser.**

**DHCP (Dynamic Host Configuration Protocol)**

En DHCP server står og lytter på et lokalt netværk på UDP port 67 og sørger for at klienter på netværket får tildelt en korrekt ip adresse og den rigtige information om netværkstruktur, altså subnet og gateways således at klienten kan kommunikere med andre computere på netværket. Forløbet er:

1. Klient starter op, altså maskinen tændes.
2. Netkortet undersøger om der er en DHCP server på netværket.
3. DHCP server svarer og sender sin IP adresse.
4. Netkortet spørger DHCP server: kan jeg få tildelt en IP adresse
5. DHCP server svarer og sender en IP adresse

#### C5. MAC layer

What are the pros and cons of parity and CRC checks?

**MAC addresser er FYSISKE addresser** på **hardware** enheder.

Ved anvendelse af **et 4 bytes CRC-felt** (Cyclic Redundancy Check) kan **man opdage eventuelle fejl** I en modtagen pakke. Feltet beregnes på basis af hele pakken, altså **inclusive** **adresse**- og **type**-felterne.

Hvis et **netkort** modtager en datapakke med **fejl** I, bliver pakken **kastet bort.** Det er så op til afsenderen evt. At blive træt af at vente på svar og sende pakken igen.

**Parity check** er en parietets bit, der altså kun kan være **0 eller 1**. Det gælder for en **hel transmission** og betyder at hvis **parietets checket fejler**, **skal hele transmissionen gentages** og ikke kun den enkelte pakke som det er tilfældet ved CRC.

#### C6. Wireless LANs

Why does 802.11 not check for collisions? Why does it handle retransmissions?

802.11 bruger teknikker til at slippe for kollision. Derudover er det også for dyrt at bygge hardware der kan opdage kollisioner.

Den bruger en ***link-layer aknowledgement/retransmission***. Når en bruger sender en frame, så er der en risiko for at dette frame ikke når til destinationen intakt.

For at løse dette fejl problem, bruger man et link-layer acknowledgement.

1. Når destinationen modtager en frame, venter den et lille stykke tid (kaldet en SIFS) og sender derefter ***en acknowledgement frame tilbage***.
2. Hvis den ikke er blevet modtaget i et bestemt tidsrum, ***så sender den en fejl og retransmitter framen*** ved brug af CSMA/CS protokollen.
3. Hvis det stadig ikke virker, så timer den ud.