Oppgave 1

1) b Kap 7.5, s 430
2) c Kap 11.3, s 587
3) c
4) a
5) d
6) d Kap 2.5, s 50
7) b Kap 2.1, s 28
8) c Kap 2.4-2.8
9) a Kap 1.3, s 7
10) c Kap 5.2, s 297
11) a Kap 12.9, s 656
12) a
13) b
14) c
15) b

Oppgave 2

a) Se figur 4.19 side 236 i læreboka
b) Utfører dataoverføringen – genererer minneadresser og buss-signaler for hver ord-overføring og holder dessuten orden på når utføringen er over.
c) Programmet som skal ha overføringen utført, blir blokkert.
   Prosessor legger inn startadresse og antall ord i DMA-registre.
   Prosessor gir beskjed til DMA-kontrolleren om å starte overføring.
   Prosessor begynner å utføre andre program / DMA-kontroller utfører overføring.
   DMA bruker avbrudd for å signalisere at utføringen er over.
   Prosessor fortsetter utføring av blokkert program.

Oppgave 3

Dekkes sort sett av 10.3 i læreboka.

a) Startbit – bit som indikerer at overføring starter. Typisk 0, mens 1 overføres når linja ikke er i bruk.
   Databit – selve dataene som skal overføres, bit for bit.
   Paritetsbit – feiloppdagende kode, for eksempel 1 hvis dataene inneholder et odde antall 1'ere.
   Stoppbit – Ekstra bit (1'ere) som skiller to overførte bytes fra hverandre.
b) Fordeler med seriell: Høyere klokkerate, tåler lengre avstander, billigere kabler og kontakter, mindre risiko for crosstalk.

Oppgave 4

1) DA – 011 ; R3 som målregister
   AA – 110 ; R6 som det første argumentregisteret
   BA – 101 ; R5 som det andre argumentregisteret
   MB – 0 ; Register, ingen konstant
   FS – 00011 ; Skal ha F = A + B + 1
   MD – 0 ; Skal skrive resultat av ALU-op, ikke minneop.
   RW – 1 ; Skal oppdatere register

2) 111 – DA ; R7 som målregister
   010 – AA ; R2 som argumentregister
   xxx – BA ; Bruker kun ett argument
   x – MB ; Bruker kun ett argument
   00001 – FS ; Operasjonen F = A + 1
   0 – MD ; Skal skrive resultat av ALU-op, ikke minneop.
   0 – RW ; Skriver ikke til register

   Konklusjon: R7 = R2 + 1, bortsett fra at R7 ikke blir skrevet til. Vil mao bare oppdatere statusregistre.

3) MOV R1, #45 - bruker ”immediate” adressering for å legge tallet 45 inn i R1.

4)
```
PUSH R4          ; Legger 8 øverst på stakken
MOV R5, #antall  ; Legger 4 (antall) i register R5
MOV R7, [R5]     ; Legger 28 i R7 – 28 ligger i minneadr. gitt av R5 (4)
POP R5           ; 8 fjernes fra øverst på stakken og legges i R5
MOV R6, 1(R5)    ; Legger 7 i R6 – 7 ligger i minneadr. 9, 1 + 8 (fra R5)
MOV R7, 10       ; Legger 45 i R7 – 45 ligger i minneadr. 10.
```

Oppgave 5

for (i = 0; i ≤ n; i++)
    a[i] = b[i] + i;

Der $3 inneholder adressen til a[0], $4 inneholder n og $5 inneholder adressen til b[0].

Bruker $1 til i og $2 til midlertidig lagring

```
        bxor    $1, $1, $1          ; Legger 0 i $1
loop:   jgt     $1, $4, slutt       ; Avslutter når i > n
        load    $2, $5, $1          ; Leser inn b[i]
        add     $2, $2, $1          ; b[i] + i
        store   $2, $3, $1          ; Skriver til a[i]
        inc     $1                  ; Øker i med 1
        jmp     loop
slutt:
end loop.
```