

Oppgave 1

$$\text{AKSESSTID} = C + (1-W) \cdot m$$

d) Utgangspunkt:

$$0,05\mu s + 0,2 \cdot 1 = 0,25\mu s.$$

$$\textcircled{1} \quad 0,05\mu s + 0,05\mu s = 0,1\mu s.$$

$$\textcircled{2} \quad 0,05\mu s + 0,2 \cdot 0,15\mu s = 0,15\mu s.$$

Alternativ 1 gir best ytelsesforbedring.

b)

Dynamisk minne er ikke like raskt, men det bruker lite avtal sammenlignet med statisk RAM. Bruker også mindre strøm enn SRAM.

c) Moores lov sier at i nyae tid er det slik at ca. hver 18. mån døbles antall transistorer på chipper. Eksponensiell vekst.

d) Nei, de kan være forskjellige.

e)

① Frekvens: $\frac{1}{16\text{ns}} \cdot 1000 = 62,5 \text{ MHz}$

② Frekvens: $\frac{1}{10\text{ns}} \cdot 100 = 100 \text{ MHz}$

③ Frekvens: $\frac{1}{8\text{ns}} \cdot 1000 = 125 \text{ MHz}$

Steg ③ har høyest ILP.

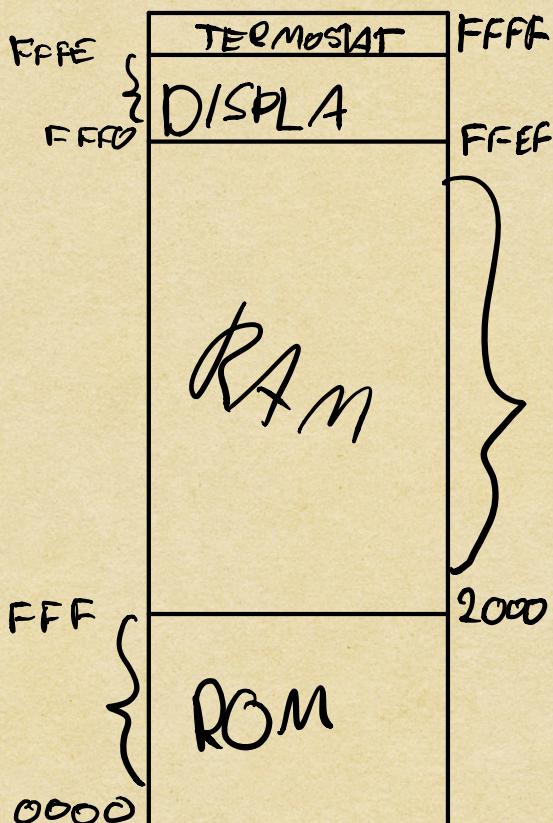
Oppgave 2.

a)
i)

ROM:

000 X X V V X X X X X X X
[0000, 1 FFF]

Ikke relativt stortes-
forhold.



Termostat:

[FFFF]

Display:

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 FFFF
[FFFD, FFFE]

RAM:

$\times \times 1 x \quad x \times \times \times \quad \times \times \times 0 \quad \times \times \times \times$
 [2000, FFFE]

ii) $2^6 = 65536$ bit

$$\frac{65536 \text{ bit}}{8} = \underline{8192 \text{ byte}}$$

b) i) 15 adresser fra FFF0 til FFFE

$$(FFE - FFO + 1) = \underline{15}$$

ii) Ingen overlapp

c) i) Nei. Det er ikke meir plass i minna. Alle adressi er brukt opp

Nila dialekten!

ii) Ja. Det er ingen Interrupt Request-linje. Oppdatering skjer kun ved polling.

Oppgave 3

a) Det niente bit'et i OPC angir hvorvidt man skal sette et hopp eller ikke.

Bit'et gjør det man kan velge mellom to programstier.

b)

1. ALU: B, C: H, MEM: 000, B: TOS

2. ALU: A+B, C: H, MEM: 000, B: LV

3. ALU: A+B, C: TOS, MEM: 000, B: OPC

Ork ikke skrive Bit-pattern. //

c)

1: ALU: 010100 (B), C: 1000000000 (H), MEM: 000, B: 0110 (nr)

2: ALU: 001100 (AND), C: 0010000000 (TOS), MEM: 000, B: 0100 (SP)

$$TOS = CPP \text{ AND } SP$$

$$\begin{array}{r} 0x5555 \\ \times 4444 \\ \hline 0x0000 \end{array}$$

$$N=0, Z=1$$

$$= \underline{\underline{0x0000}}$$

Oppgave 4

- a) Dette er en RISC-prosessor fordi det like lange instruksjoner, og de er meget enkle. Det er oppgitt at det er mange generelle registre. I tillegg er det load/store-arkitektur.
- b)
- i) En-adress-instruksjon. Bruker kun adresse på den skal lagre verdien.
 - ii) MOVC bruker immediate addressing konstant/data sitt som operand.
 - iii) Register addressing.

c) Ja. Vi har 8 bit for å lage adresser. 16 adresser krever kon 4 bit ($2^4 = 16$), så vi kan få potensielt $2^8 = 256$ adresser.

ii) Ja. Har branching, VO og ALU-funksjoner.

d) STORE R₁, R₃

$$[0xAAAA\ AAAAA] \approx R_1 = 0x0000\ AAAA$$

STORE R₂, R₄

$$[0x5555\ 5555] = R_2 = 0x0000\ 5555$$

INV R₃, R₅

$$R_3 = 0x5555\ 5555$$

INV R₄, R₁

$$R_4 = 0x\ AAAA\ AAAA$$

LOAD R₁, R₃

$$R_1 = [R_3] = 0x0000\ 5555 \quad \leftarrow$$

LOAD R₂, R₁

$$R_2 = [R_1] = 0x0000\ AAAA \quad \leftarrow$$

i) STORE R₁, R₃

$$[0x\ AAAA\ AAAA] = 0x0000\ AAAA$$

STORE R₂, R₁

$$[0x5555 \ 5555] = 0x0000 \ 5555$$

CP R₁, R₂

$$R_1 = 0x0000 \ 5555$$

CP R₂, R₁

$$R_2 = 0x0000 \ 5555$$

Det samme, idet
likt som resultat
over