5.4:’

4. Given the memory values below and a one-address machine with an accumulator, what values do the following instructions load into the accumulator?

word 20 contains 40
word 30 contains 50
word 40 contains 60
word 50 contains 70

a. LOAD IMMEDIATE 20
b. LOAD DIRECT 20
c. LOAD INDIRECT 20
d. LOAD IMMEDIATE 30
e. LOAD DIRECT 30
f. LOAD INDIRECT 30

6. Compare 0-, 1-, 2-, and 3-address machines by writing programs to compute

$$X = (A + B \times C) / (D - E \times F)$$

5.6:

0 Address	1 Address	2 Address	3 Address
PUSH M	LOAD M	MOV (X = Y)	MOV (X = Y)
POP M	STORE M	ADD (X = X+Y)	ADD (X = Y+Z)
ADD	ADD M	SUB (X = X–Y)	SUB (X = Y–Z)
SUB	SUB M	MUL (X = X*Y)	MUL (X = Y*Z)
MUL	MUL M	DIV (X = X/Y)	DIV (X = Y/Z)
DIV	DIV M		

Address 0:
Push A
push B
push C
MUL (B*C)
add (a+B*C) - fordi vi har omvendt polsk notation
push D
push E
push F
mul
sub
div
pop X

Address 1:

LOAD E
MUL F
STORE (E*F)
LOAD D
SUB (E*F)
STORE (D-E*F)
LOAD B
MUL C
ADD A
DIV (D-E*F)
STORE (X) (Måske? prøvede først med a+B*C, men brugte flere store end nødvendigt, så forkortede til sådan her version)

Address 2:
MOV R0, B
MUL R0, C
ADD R0, A - - R0=B*C+a
MOV R1, E
MUL R1, F
MOV R2, D
SUB R2,R1 - - R2= R2-R1
DIV R0,R2 - - R0/R2
MOV X, R0

Jeg tror muligvis man kan spare på mængden af registre, ved at ændre ordren, ligesom ved ADDRESS 1, fordi så må man kunne genbruge det registre der bruges til at gemme E*F?

ADDress 3:
MUL R0, B, C
ADD R0, R0, A
MUL R1, E, F
SUB R1, D, R2
DIV X, R1, R2

For 0 address
Opcodes = 8 bits, og vi har 5 brug af disse, så 5*8 = 40
push og pop er vel opcodes + address så 8+16 = 24, vi har 7 af disse.
7*24=164, alt i alt 204 bits.

For 1 address
Alle instruktioner er opcodes + address, så 8+16=24
der er 11*24= 264 bits

For 2 Address
Der er vel forskellige kombinationer af registre og memorty
I vores tilfælde har vi

For 3 Address
2*(2*memory+registre+opcode)=2*(32+4+8)=88 bits
3*(memory+2*registre+opcode)=3*(16+8+8)= 92 bits
92+88=180

5.14

14. How many registers does the machine whose instruction formats are given in Fig. 5-24 have?

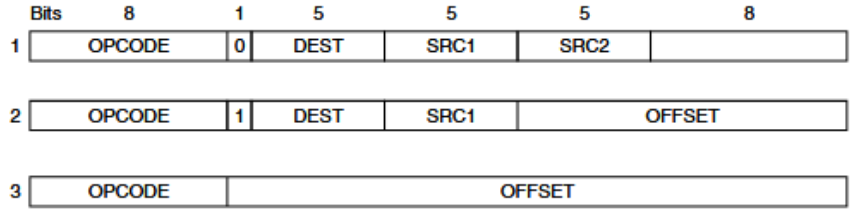


Figure 5-24. A simple design for the instruction formats of a three-address machine.

Hver af disse "registre" peger jo på en register id. Lad os sige at DEST har værdi 00011, det betyder den peger på værdien inde i register 3. Men de deler samme registre så vi får
2^5 = 32

5.21
Vi skal vel have ændret værdi til 0 i memory eller register
Det kan vi vel gøre ved at trække indholdet fra sig selv.
Alternativt
LOAD 0 ind i register
Alternativt XOR et tal med sig selv. XOR giver kun 1 hvis A og B er modsat hinanden

5.23
b = a xor b
a = a xor b
b = a xor b

a = 01

b= 10

b = 01 xor 10 = b = 11
a = 01 xor 11 = a= 10
b = 10 xor 11 =b= 01
Ergo vi har lige vendt rundt på de to variabler

Det er kommet frem til ved at studere logik tabellen, da de indses at a og b kan flippe, men output evare det samme. Vi kan se det i følgende:
Hvis vi siger
a = A
b = B

og vi siger
b = a xor b
a = a xor b
b = a xor b

Det vil sige
b = A ⊕ B
a = A ⊕ (A ⊕ B), Associative lov: (A ⊕ A) ⊕ B, A ⊕ A = 0, ergo, 0 ⊕ B = B. Så a = B
b= B ⊕ (A ⊕ B), Kommutative lov: B ⊕ (B ⊕ A), Associative lov: (B ⊕ B) ⊕ A => 0 ⊕ A.
Det vil sige b = A
Ergo har vi vist hvorfor det dur.

Exercise part 2:

a) What are the opcodes for JMP, PUSH and POP?

JMP = 1F

PUSH = 32

POP = 36

b) Is the opcode the same for all adressing modes of MOV?

No, der er

MOV reg, reg (06)

MOV reg, address, (03)

MOV address, reg (05)

c) How are the registers (A - D) addressed?

As simple letters, but can also be addressed as [x], where that actually means the address using a general purpose (gp) register.

A = 00

B= 01

C= 02

D = 03

d) The program counter in this machine is called IP (instruction pointer) can Schweigi use program counter relative addressing?

No, arithmetic can not be performed on the instruction pointer

e) Which of the addressing modes mentioned in todays slides are available in Schweigi (look in the instruction set documentation)

Immediate addressing MOV reg, constant

Register direkte addresering MOV reg, reg

Indirect addressing med en signed integer fra -16 til 15

Dermed også Indexed Addresing

Stack PUSH, POP

f) Consider exercise 6 from above (Tanenbaum). Make a small program in Schweigi that solves the calculation so that you use as little memory as possible. The numbers A, B, C, D, E and F should be 8-bit values stored with corresponding labels in memory. The result X should be stored in a memory location labelled X.

Tog lidt langt tid

JMP start

A_VAL: DB 10

B_VAL: DB 5

C_VAL: DB 3

D_VAL: DB 20

F_VAL: DB 4

E_VAL: DB 2

```
X:     DB 0
store: DB 0
store1: DB 0

start:
; Beregning af (d-e*f)
MOV A, [E_VAL] ; Det skal være A-registre fordi vi kan kun lave * i A
MUL [F_VAL] ; e*f
MOV [store], A
MOV B, [D_VAL]
SUB B, [store]
MOV [store], B

MOV A, [B_VAL]
MUL [C_VAL]
MOV [store1], A
MOV B, [A_VAL]
ADD B, [store1]
MOV [store1], B

MOV A, [store1]
DIV [store]
MOV [X], A

HLT
```

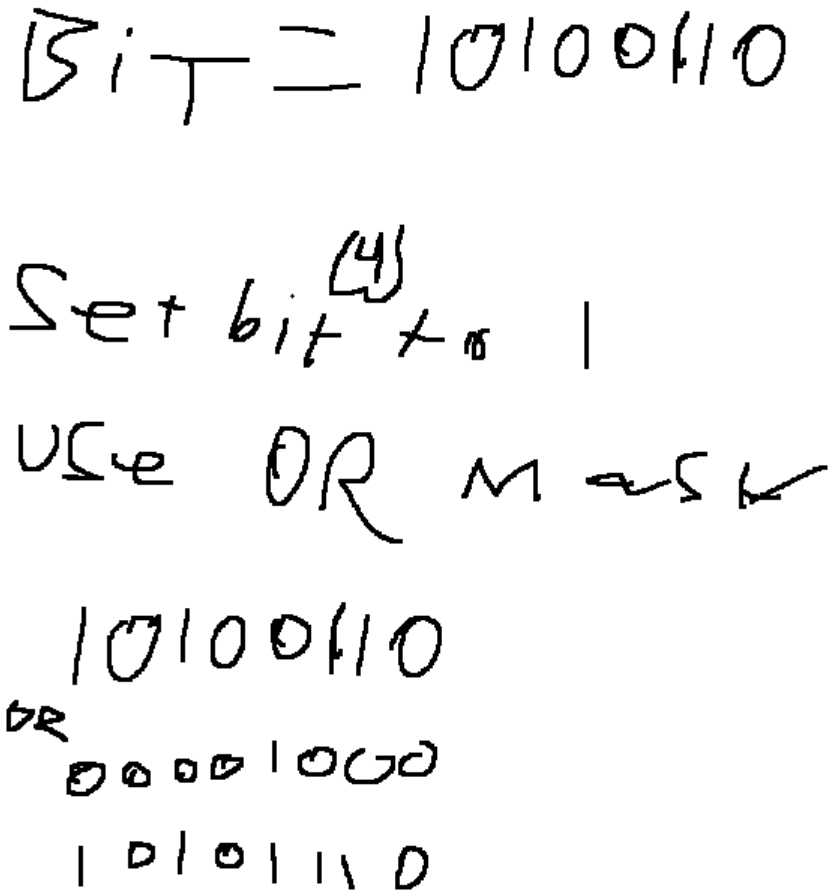
Der er klammer ved alle labels, i det jeg mener hvis ikke man har dem, så er addressen bare værdien af variablen, men hvis jeg bruger klammer, så bruger den faktisk værdien i variablen.

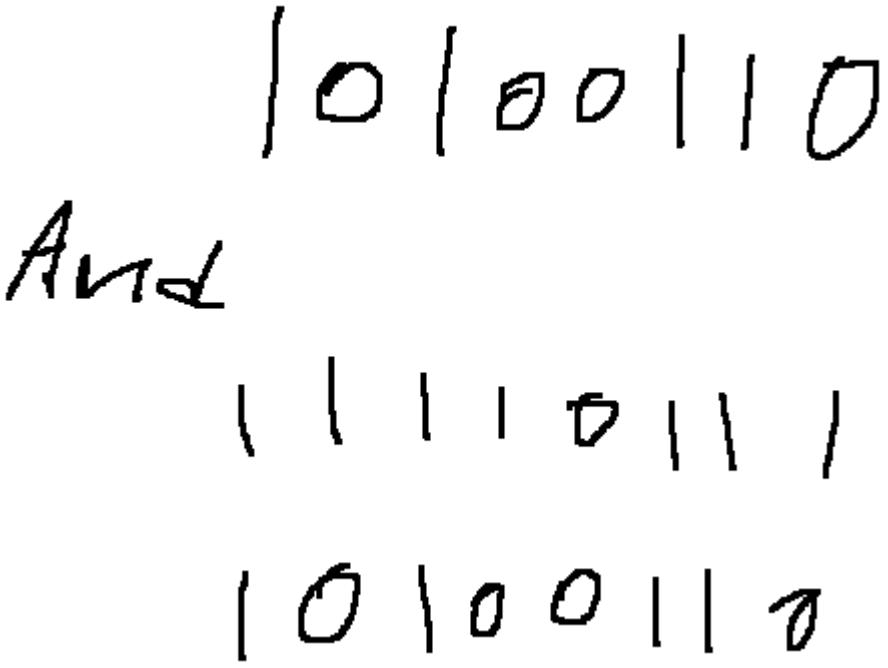
g) Schweigi has no bit manipulation commands. Find a way to: set a certain bit in a register to 0 or 1 and a way to toggle one bit in a register between 0 and 1. The operation should not change the rest of the bits in the register. (Hint: Logic)

Bit masks nok med forskellige logic, og bitmask

vi kan bruge OR til at ændre 0 til 1, AND til at ændre 1 til 0, XOR til at flippe.

eksempel i paint:





Vi inverser bit mask

Til bare at toggle frem og tilbage, kan vi bruge xor, hvor bitmasken er 1 der hvor bitten skal ændres, fordi når man xor med 0, forbliver det altid det originale, f.eks: 0 xor 0 = 0, 0 xor 1= 1