**Innledning**

Chip 8 ble først tatt i bruk av COSMAC VIP og TELMAC 1800 8-bits Mikro-komputere på midten av 1970-tallet. Chip 8 programmer blir kjørt på en chip 8 virtuell maskin. Grunnen til at den ble laget var for å gjøre spill enklere å programmere på tvers av datamaskiner. Chip 8 kjører mange klassiske spill som for eksempel Pong, Space Invaders, Tetris og Pac-man. Vi arbeidet sammen for å få frem produktet og møtte opp til avtalte timer for å få gang i sakene. Vi hadde mange spørsmål innad i gruppen slik som, hva er en chip 8 emulator egentlig? Hvordan laget folk spill til den og hva kan vi lære fra det? Dette er spørsmål som ble svart relativt kjapt i arbeidsprosessen og det bringte oss videre til å utvide egenskapene til Chip-8.

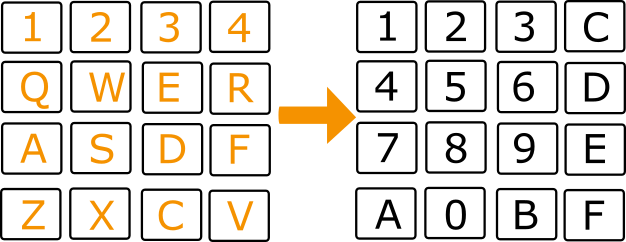
**Chip 8 Emulator**

En emulator er et dataprogram som lar oss etterligne designet og funksjonaliteten til et datasystem (System X). Emulatoren lar oss kjøre software ment for System X på et helt anderledes datasystem (System Y). For eksempel så kan vi kjøre en gameboy emulator (System X) på vår lokale PC (System Y) for å spille gameboy spill.

Chip 8 emulatoren vi lager etterligner maskinen som ble laget på midten av 1970-tallet og lar oss kjøre Chip 8 spill på pcen vår via emulatoren. En chip 8 er bygget opp av 16 8-bits registeret ved navnet V0 til VF. Chip 8 har også to timers som teller ned på en felles 60 hertz klokke, helt til de når 0. De forskjellige timerene er:  
  
Delay timer: Denne brukes til timing-relaterte hendelser i spill. Verdien kan settes og leses.  
Sound timer: Denne brukes for lydeffekter. Når verdien ikke er 0 så lager den en pipelyd.

**Tastatur og Grafikk**

Chip 8 bruker et heksadesimalt tastatur med 16 taster som går fra 0 til F. Tastene ‘8’, ‘4’, ‘6’ og ‘2’ blir typisk brukt for bevegelse. Tre operasjonskoder blir brukt til å detektere knappetrykk. Vi går mer inn på operasjonskoder i neste del av rapporten. I emulatoren vår brukte vi keymappet under:

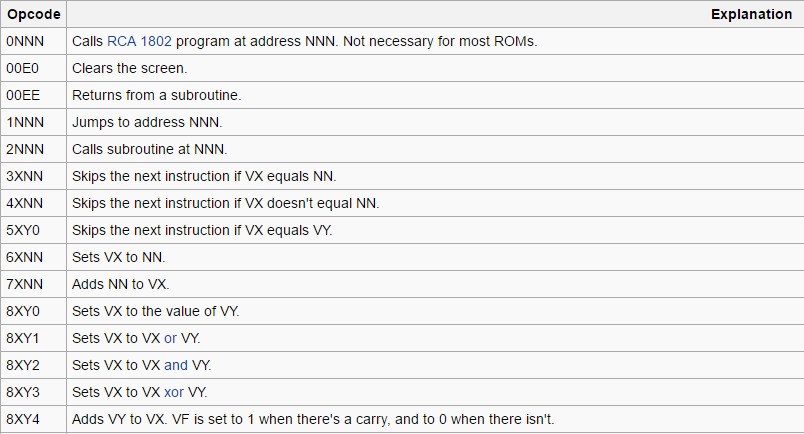


*(Figur X.1) Utsikt over knappene vi bruker på tastaturet i forhold til hva de er på Chip-8 sitt Hex Tastatur.*

Et originalt chip 8 display er 64x32 pixler og har monokromt fargepalett. I vår chip 8 la vi til ekstra farger via biblioteket OpenGL. Grafikken i chip 8 blir tegnet med noe som heter ’sprites’. Sprites er to-dimensjonale figurer som blir satt sammen til å vise forskjellige objekter. I chip 8 tegner vi inn på skjermen for å få sprites.

**Operasjonskoder**

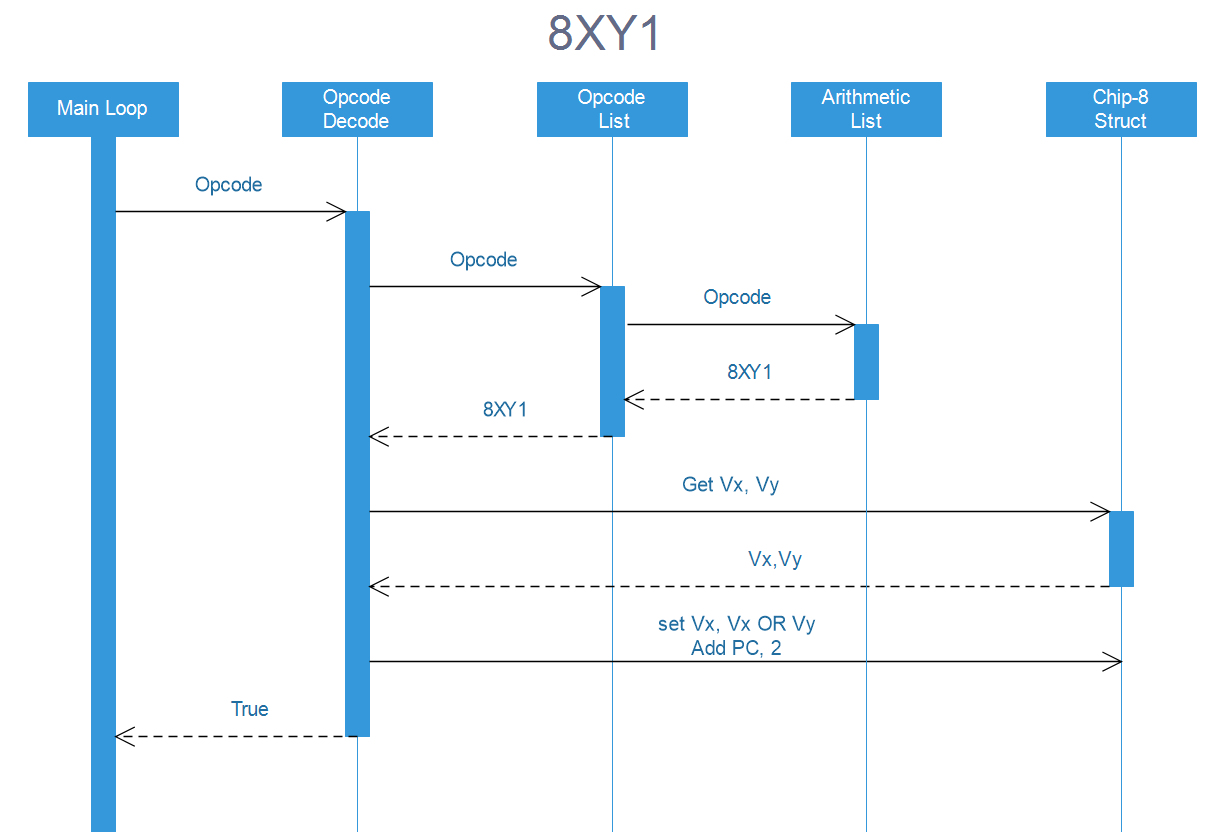
En operasjonskode er delen av en maskinkode instruksjon som spesifiserer hvordan en operasjon skal utføres. Instruksjonene spesifiserer vanligvis dataen de vil arbeide med, i form av mattematiske operasjoner. Chip 8 har 35 forskjellige operasjonskoder som styrer alle funksjonene til maskinen.

*(Figur X.2) En liste over de 15 første operasjonskodene i Chip 8.*

Operasjonskoder er delt opp i forskjellige hovedgrupper. Disse gruppene er sortert etter funksjonaliteten operasjonskodene har. I figur X.2 så ser vi for eksempel at at opkodene 3XNN til 5XY0 omhandler samme type operasjoner, men med forskjellige parametere. For eksempel, operasjonskoden 5XY0, når kjørt, hopper over neste instruksjon hvis register VX er det samme som register VY.

**8XY1 – Sekvensdiagram**

Nå som vi har litt forståelse om hvordan en Chip 8 emulator er bygget opp, så ser vi på et sekvensdiagram for en operasjonskode. Ut ifra figur X.2 ser vi at 8XY1 har beskrivelsen: Sets VX to VX or VY.

*(Figur X.3) Sekvensdiagram for operasjonskoden 8XY1*

Vi ser for oss at VX = 10110011 og VY = 00111010 før denne operasjonskoden tar sted. Main loopen sender opcode signalet videre til dekoderen. Dekoderen sender også opcode videre hvor da opcode listen detekterer det første tallet. Ut ifra hva det første tallet er (I dette tilfelle 8) så sendes den videre til hvor den hører til. I dette tilfellet er det aritmetic blokka fordi vi skal utføre en aritmetisk operasjon. Her detekterer vi det siste tallet for å signalisere hvilken operasjon vi skal utføre så sender vi opcode tilbake til dekoderen. Dekoderen tar da og sender ut X og Y verdiene (bitverdiene) som skal inn på registrene. Her detekterer vi hvilke registre som verdiene skal puttes inn i. Til slutt utfører vi operasjonen og setter VX til å være VX or VY. I vårt tilfelle blir det 10110011 OR 00111010 som gir oss resultatet 10111011. Når verdien er på plass så sender den true tilbake til main, som signaliserer at operasjonen er utført.