kunne lage et detaljert design på bit-nivå for et datasystem i cyberdyret med en basismodell for PROSESS ADMINISTRASJON og MULTIPROGRAMMERING;

lage språk i bits som dataen “forstår” i form av bits, og bestem hvordan data skal bli overført? vi bestemmer selv hvordan kode blir tolket

skriv et språk! forklar meningene med ordene

feilkode system? debugging : hvordan skal dyret reagere om en feil skjer

Som illustrert i tidligere versjon av cyberdyret, har vi en prosessor ment for å håndtere og koordinere oppgavene til dyret. Prosessoren er sammenkoblet med en minnecelle, som fungerer som både register og korttidshukommelsen til dyret. Registeret er det som gir dyret evnen til å tolke mottatt kode, og derav vite hvor og hva som skal initialiseres.

Korttidshukommelsen tillater dyret å motta “befalinger”, for så å lese over og “holde på” det når hele koden er sendt. Hvis dyret ikke hadde hatt noen form for mellom-minne ville ikke dyret kunne håndtert datastrømmen og oppgave initiering samtidig.

Selve dataen dyret vil motta er basert på sekvenser med åtte signaler (bits) av gangen; en byte. I de åtte bittene representerer de første fire en lokasjon, mens resterende representerer en handling. Fire bit gir 15 alternativer, så vi har 15 diverse befalinger, mens vi kun har fire destinasjoner, altså masse ubrukt plass. Vi kan derfor vurdere å anvende rekker med tre bit destinasjoner og fem bit instrukser. En oppsette ser slik ut:

Binær kode i registeret

Kode

/symbol

Beskrivelser

0000 0000

{

Start (startpunktet til koden/loopen; hva som skal repeteres).

0000 1111

}

Slutt (stopp av kode/loop).

…. 0001

Loop

Loop (loop er alltid lik 4).

"{ }" kan brukes for å definere en loop inn i en kode

…. 0010

minus | -

Summer variabel X med “-1”.

…. 0011

pluss | +

Summer variabel X med “+1”.

…. 0100

X

Setter verdi for X (Kun hele tall).

…. 0101

Roter

Roter (roter begge hjulene 0.5 rotasjoner) 90 grader med klokka.

…. 0110

Frem

Frem (roter begge hjul \*X rotasjoner).

…. 0111

;

Hver befaling skilles med ";".

“;” står også for "gå videre".

…. 1000

if

en blokk med kode som kan utføres om betingelsene er sanne.

…. 1001

else

Sjekker om betingelsene til if er sanne eller ikke og utføres på grunnlag av det.

…. 1010

&&

returnerer sann hvis begge uttrykkene er sanne.

…. 1011

Sammenligner to verdier, “>” = større enn.

…. 1100

<

Sammenligner to verdier, “<” = mindre enn.

…. 1101

<=

Sammenligner to verdier, “<=” = Større enn, eller lik.

…. 1110

=

Sammenligner to verdier, “<=” = Mindre enn, eller lik

….

=

Sammenligner to verdier, “=” = er lik

Koden tolkes ved at dyret har en intern høyfrekvent timer som ved jevne intervaller forteller prosessoren at den skal lese om i hvilke grad den mottar elektrisitet (amplituden på signalet), og videresender denne informasjonen til mellomlagringen. Ved lavt signal mottar dyret “0” mens ved høyt signal tolker dyret “1”.

Binær kode

Destinasjon

Beskrivelse

0001 ….

Prosessor

Sender data til dyrets “hjerne”

0010 ….

ROM

Sender data til ROM

0011 ….

RAM

Sender data til RAM

0100 ….

Hjul 1 frem

Roter hjulet med klokken

0101 ….

Hjul 1 bak

Roter hjulet mot klokken

0110 ….

Hjul 2 frem

Roter hjulet med klokken

0111 ….

Hjul 2 bak

Roter hjulet mot klokken

Enhver lokasjon i dyret har sin egen adresse, dette er for å slippe at enhver celle skal ha sine egne veier kun mellom seg selv og prosessor. De fleste celler bruker derfor samme to linjer; en fra prosessor og en til prosessor. Dette er simpelthen for å spare plass i dyret. Dette medfører likevel noen komplikasjoner, nemlig at flere celler ikke kan sende data samtidig og at cellene må kunne vite når hvilke data er ment for dem. For å hindre at cellene “snakker i munnen på hverandre” har hver enkelt destinasjon et fast tidsrom hvor de kan sende signaler. For at en celle skal vite om den blir adressert vil alle cellene måtte ha muligheten til å “lese” hva som blir kjent så de har alle en svak prosessor som leser strømmen, og ser etter sin spesifikke adresse.

Hjulene vi bruker en noe annerledes, da vi trenger de til å rotere i begge retninger for at dyret skal kunne rotere. Vi har derfor tatt inspirasjon fra hvordan en DC elektromotor ville fungert, og derav valgt å gitt hvert hjul to inputs; en som bestemmer om det skal rotere med klokka, og en mot klokka. Dette gjøres ved å tillate energien (elektrisiteten) dyret innehar til å gå i en av to retninger, basert på hvilke input som blir aktivert.