INF2270 – Obligatorisk oppgave 2
”En liten CPU”

Hovedmål
Denne obligatoriske oppgaven skal dere bruke DigLog verktøyet med simulator til å implementere og teste en liten CPU som bruker 4 maskinkode instruksjoner.

1. Oppgaven

Følgende deloppgaver vil bidra til en strukturert gjennomføring av oppgaven.

- Dere skal bruke en arkitektur som har separat program minne og data minne (kalles Harvard Architecture). For minne elementene kan dere bruke cell SRAM8K. Ettersom vi kun skal bruke 16 adresser (altså 4-bit) av minneelementet så kan dere jorde **(GND)** alle adresse innganene **(PIN)** på venstre side bortsett fra de 4 øverste. Sett **”mode”** i celle konfigurasjonen til **”read-only”** for program minne, mens for data minne setter dere **”mode”** til **”deposit”**. (for å åpne konfigurasjons menyen/muligheten for en celle, må dere traversere gjennom de ulike **”modes”**(ene) i nedre høyre hjørne av programmet (ROT, MIR-, MIR|, CNFG) til mode er satt **CNFG**. Deretter kan dere venstre-klikke på cellen i designet, så vil konfigurasjonsmenyen åpnes seg i en seperat vindu i Linux og i samme vindu i Windows miljøet). Det å bruke Harvard Arkitektur med dedikert instruksjonsbus og små raske minneelementer har følgende fordeler:
  - Man trenger egentlig ikke en instruksjons register (IR), men kan bruke utgangen **(output)** av instruksjonsminne direkte.

- Anta at ingen av de 4 instruksjonene implementere en **jump** og alle instruksjonene kan bli eksekvert innen en klokkesykel **(single clock cycle)**. Dette vil medføre en rekke konsekvenser, der i blant:
  - Vi kan faktisk bruke en teller **(counter)** celle (for eksempel 16COUNT) som instruksjonspeker/program teller (IP/PC) som teller opp hele tiden uten avbrytelser **(interrupt)**. Konstruer telleren på en slik måte at en **pulse switch** kan resette tellere og bruk kombinatorisk logikk til å unngå at telleren teller mer enn til 16.
  - Ettersom alle instruksjonene blir ferdig innen en klokkesykel så trenger vi ikke å ha en tilstandsmaskin som blir trigget av instruksjoner. Enkel kombinatorisk logikk vil holde i dette tilfellet.

- Det holder med en 8 bit data register til CPUen. Vi kaller den for **R.** Bruk gjerne celle **74374** for dette registeret.
-
- Det skal være en **single system bus** (merket som S7….S0) i tillegg til instruksjons bus(en) som kobler sammen data minne, (deler av) instruksjons bus(en) og utgangen **(output)** av **R** til inngangen av

dataregisteret **R.** Dere må sørge for at det kun er **en** aktiv kilde **(source)** som driver hver enkelt linje **(line)** av denne bus(en), slik at alle kilder som potensielt kan drive en linje er tilkoblet en **"tri-state".** Merk at minneelementet SRAM8K har en tri-state utgang som er kontrollert av pin **OE**. Men for å koble instruksjons bus(en) og utgangen av R til inngangen av R så burde dere bruke celle **74244** med to sett av 4 bit tri-state buffere.

- De fire 8 bit instruksjonene er beskrevet i tabellen under. "DDDD" i maskin koden referer/menes "data", mens "AAAA" er data i minneadressen.

Forklaring	Maskinkode
Load de 4 **LSB**(ene) av R med DDDD	1000DDDD
Load de 4 **MSB**(ene) av R med DDDD	1001DDDD
Load R fra dataminne innhold gitt adressen AAAA	1010AAAA
Write R til dataminne innhold gitt adressen AAAA	1011AAAA
Alle andre operasjonskoder skal være **NOP** (no operation = ikke gjør noe)	

Dere kan **load**(e) programmer inntil 16 instruksjoner in på program minne ved å åpne minne konfigurasjonen i DigLog. Bruke piltastene til å gå til **"File name to save:".** Skriv så inn filnavnet og trykk på ENTER-knappen. Dette vil lage en ASCII fil med minneinnholdet i en passende format. (Dere kan også bruke denne metoden til å sjekke tilstanden av deres data minne etter en programkjøring.) Bruk så en teksteditor til å editere den første linjen i filen. Retningslinjen/ konvensjonen er at de laveste adressene er to **bytes** (hexadesimale) helt til venstre, med de 4 MSB(ene) til høyre og de 4 LSB(ene) til venstre. Slik at følgende linje:

0000:8597B18194A10000000000000000000000

blir eksekvert slik:

85
97
B1
81
94
A1

HINT! I forhold til å kunne fullføre alle instruksjonene innen en klokkesykel, burde dere bruke **falling edge** av klokkesignalet til å trigge **loading**(en) av data registeret **R.** Merk dere også at flip-flopen i cellen SRAM8K er transparant latch som er kontrollert av pin R; hvis R er lav og OE er høy så er minne gitt av adressen bestemt av / gitt av adresse pinnene lastet inn **(loaded).** Hvis data inngangen endres når R er lav så vil dette også endre innholdet av minne elementet.

2. **Verktøyet**
   Dere har nå hatt anledning til å knytte bekjentskap til DigLog i Obligatorisk oppgave 1. Hvis dere ønsker mer dokumentasjon kan dette finnes her:
   http://www.cs.berkeley.edu/~lazzaro/chipmunk/

3. **Innlevering**
   Innleveringen skal skje gjennom en skriftlig rapport hvor dere inkluderer **plot** av deres krets og **signal trace** fra simuleringen. Det er viktig at dere følger retningslinjene deres gruppelærer gir.

   Viktig at dere også leverer inn design filen (.lgf filen)

   Personlig vil jeg oppfordre dere alle til å gjøre innleveringen så oversiktlig som mulig slik at dere kan få gode tilbakemeldinger fra deres gruppelærer.

Lykke til ☺