Journal Dag 1

Øvelse 1.1

Den første øvelse startede med at blive fortrolig med udviklingsværktøjet ZDS II. Vi lavede et nyt projekt, hvor indstillingerne blev sat til en CPU af typen Z8F6403 og et registerminde på 4KB. Vi skrevskrevet en simpel C-fil der udskrev "hello world". Pakkene ez8.h og sio.h blev også inkluderet i filen. Til sidst blev C-filen gemt, bygget og simuleret i HyperTerminal.

Øvelse 1.2-1.3

I denne øvelse blev der opgivet en C-fil der indeholdt fejl. Motivationen for dette, var at lære hvordan ZDS II's debugger fungerer. Debuggeren blev brugt til at manuelt køre linje for linje i C-programmet helt til alle fejl var rettet. Debuggerens "Go to Cursor" og "Step Over" blev flittigt brugt sammen med en oversigt over variablene i og r. En af de fejl der blev opdaget var at funktionen power multiplicerte tallet a en gang for meget. Dette medførte bit-overflow, og forkerte udregninger som følge. Denne fejl blev rettet i for løkken, og programmet kørte korrekt.

Øvelse 2

I denne øvelse blev HyperTerminal's output kontrolleret ved hjælp af nogle selvdefinerede funktioner. Vi lavede alle de funktioner der blev bedt om og supplerede også med de frivilligefunktioner(up, down, left, right). Funktionen clrscr() som renser skærmen. funktionen clreol() som renser resten af linjen som markøren står på. Funktionen gotoxy(unsigned char x,unsigned char y) som flyttede markøren til den valgte position. Funktionen underline(char on) som bruges til at vælge eller fravægle om teksten skal være understreget. Funktionen blink(char on) som gør, at den skrevne tekst blinker. Funktionen reverse(char on) som bruges til at bytte om på farvene på forgrund og baggrund.

Til sidst lavede vi en funktion der kunne lave et vindue i terminalvinduet. Funktionen accepterede to sæt med koordinater(for hhv. Nordvestlige og sydøstlige hjørner), stilform for vinduet og titeltekst. Vi besluttede at lave funktionen således den ikke skrev de blanke mellemrum inde i vinduet.
På denne måde kunne man omringe tekst der allerede var skrevet. Funktionen blev udstyret med 2 forskellige stilarter, som gav forskelligt udseende vinduer.

Journal Dag 2

Øvelse 3

I denne øvelse startede vi med at lave en filstruktur vi kunne bruge senere. En header-fil blev defineret med de funktioner der blev skrevet under sidste øvelse. For at systemet skulle køre effektivt blev denne header-fil bliver kun læst ind hvis den ikke var læst ind fra før. Et nyt projekt blev lavet og de gamle funktioner blev gemt i filen ansi.c. De funktioner som beskrev fixed-point aritmetik blev skrevet i filen math.c.

Øvelse 3.1

Denne øvelse, der indeholt spørgsmål omkring bit-manipulation blev ligeledes besvaret.

Øvelse 4.1

Instruktionerne blev blot fulgt og vi fik med succes skabt en LUT vha. excel-programmet, excellut.

Øvelse 4.2

Vi skulle i denne opgave lave en funktion der kunne finde sinus for enhver integer som input – den skulle således kunne finde for negative inputs og positive inputs. Vi kunne i denne sammenhæng blot bruge to complement's repetetive natur sammen med sinus repetetive natur, og derved indse at vi blot kunne fjerne de sidste 7 bit, og kun bruge de første 9. Dette blev gjort ved en simpel bit operation, hvor inputtet blev and'et med 0x01FF, hvilket gjorde, at vores input altid var mellem 0-512.

Cosinus funktionen implementeres også ved blot at bruge egenskaben cos(a)=sin(a+90)
Da vi testede den funktion fandt vi et problem med compileren, da den compilerede -128 som 128. Følgende workaround blev brugt: -128 = -(int)128

Øvelse 4.3

I denne øvelse skulle vi lave en funktion der kunne rotere en vektor.
Vi lavede første en funktion, der kunne initialisere en vektors x og y værdier. Disse inputs var floating point format, og blev omformet til fixed-point format.
Rotationsfunktionen blev derefter implementeret, hvor multiplikation blev gjort med de opgivne makroer og de før kreerede cos- og sinusfunktioner.

Øvelse 4.4

Denne øvelse, der indeholt spørgsmål omkring bit-manipulation blev ligeledes besvaret.

Journal Dag 3

Øvelse 5.1

I denne øvelse skulle vi lave en funktion, readKey() der kunne læse brugerinput på de tre knapper. Vi skulle derefter teste funktionen med en counter. Vores funktion virkede ikke i første omgang, da vi havde læst kredsløbs-diagrammet forkert og troet at schmitt-triggeren på inputtet var af inverterende type. Vi regner derfor med at inputværdierne var høj når de blev trykket – det var selvfølgelig ikke tilfældet og det gav os problemer – det virkede dog da vi fik det rettet. Vi havde også nogle problemer med debouncing og dårlige knapper. Alt i alt blev resultatet acceptabelt.

Øvelse 5.2

I denne øvelse skulle vi lave et program der ville outputte værdien for vores counter fra øvelse 5.1 og vise dem på LED'erne på boardet. Dette blev gjort ved at sætte værdien af counteren inputtet til LED'erne. Dette program fungerede som forventet.

Øvelse 6

I denne øvelse skulle vi lære at skrive til en timer. Vi skulle i høj grad blot følge instruktionerne. Vi valgte en prescale værdi på 2^7, altså den største mulige for timerne. Reloadværdien blev da udregnet udfra dette. Vores nederste værdi blev bare sat til 1. Interrupt prioriteten blev sat til normal. Vi havde nu problemer med at forbinde til vores board, men simuleringen i programmet fungerede fint. Dagen efter fik vi vores board til at fungere igen, og så at vores program fungerede.

Øvelse 6.1

Vi brugte vores timer fra øvelse 6 til at lave et stopur, og det fungerede også som forventet. Vi brugte her knapperne på boardet til at styre vores stopur. Det var her vigtigt at gøre så lidt som muligt i vores interrupt-funktion, da vi ønsker at den skal være hurtig. Vores samlede stopurs løsning brugte funktionen windows(unsigned char x, unsigned char y, unsigned char x1, unsigned char y2) til at danne rammen. Derefter blev der lavet en funktion som returnerer tiden fra vores timer. Dette blev brugt til at skrive splittime 1 og 2. Derefter blev tiden skrevet sådan, at den ville opdaterer hvert sekund. Programmet blev lavet sådan, at det også var muligt at resette timeren.

Journal Dag 4

Øvelse 7

I denne øvelse skulle vi skrive en streng indeholdende 4 karakterer på vores LED-skærme. Dette gjordes ved at multiplexe signalerne ud. Først skulle vi have en ny clock med en periodetid på 0.5 ms. Vores design fungerede ikke helt i første implementation, da bogstaverne var vendt forkert. Dette skyldes at søjle 0 på LED'en er søjlen helt til højre(og 4 til venstre), og vi havde således vendt bogstaverne om. Dette blev fikset og derefter fungerede vores design efter hensigten.

Øvelse 8

I denne øvelse skulle vi have en streng til at rulle henover skærmen. Dette blev løst med en en buffer som indeholdte 5 bogstaver. Fire bogstaver som vises, og det femte som er klar til at blive hentet ind ligger i bufferen. Den måde rulle effekten blev implementeret var ved at skifte søjlerne en gang til venstre, og dermed give effekten af, at der rulles mod højre. Når der er blevet skiftet 6 gange, svarende til et helt tegn indlæses et nyt tegn i videobufferen, og det som tidligere stod først smides ud. Sådan fortsætter koden indtil den løber ind i enden på strengen. Denne øvelse blev også løst, og forskellige funktioner, som udbyggede funktionaliteten blev lavet, som blev brugt senere i projektet.