Maskinorienterad programmering

Arbetshäfte för laboration nr 4

Användning av XCC för korskompilering till MC12

Styrning av borrmaskin

Förberedelser

Observera att du måste ha arbetat med ARB1 särtryck ur ”*Arbetsbok för MC12”*, avsnitt 5, ”*Maskinnära programmering i C och assembler*” för att kunna genomföra laboration nr 4. Två laborationstillfällen är avsatta för laborationen. För att hinna göra laborationen är det viktigt att du **före** laborationstillfällena har skrivit programdelarna och helst testat dem i källkodsdebuggern i XCC.

Version 110221

Ifylls med bläck!

Laborant:

_____ _____ _____

Personnummer Namn (textat) Datum

Godkännande - laboration:

_____ _____

Laborationshandledares underskrift Datum

Se till att få ovanstående ifyllt. Det är ditt kvitto på att du är godkänd på denna laboration.

Avsikt

I denna laboration får du lära dig hur man med hjälp av en korskompilator kan utveckla C-program för en dator som direkt styr hårdvara.

Programmeringsmiljö

Korskompilatorn XCC skall användas. Du kan själv ladda ner och installera XCC på en dator som kör Windows. Du finner länken på kurshemsidan.

Förberedelser

Laborationen utgörs av fyra uppgifter och två laborationstillfällen finns tillgängliga för att lösa dessa. **Observera att denna laboration KRÄVER att du har utfört mycket av förarbetet innan du kommer till dina laborationstillfällen.** Laborationstiden i sal räcker normalt **inte** för att lösa dessa uppgifter. Du kan använda XCC's inbyggda simulator för att testa dina program i förväg.

Arbeta först igenom ARB1 särtryck ur"*Arbetsbok för MC12", avsnitt 5, sid 64-75 tom uppgift 112.* Lös därefter de fyra uppgifterna i detta lab PM.

Det är både tillåtet och lämpligt att du arbetar tillsammans med andra för att lösa uppgifterna. Vid laborationstillfällena kommer du att få demonstrera dina lösningar med såväl simulator som i laborationssystemet (MC12). Du måste också vara beredd på att besvara frågor kring dina lösningar vid detta tillfälle. När du visat upp fungerande lösningar och visat att du dessutom kan förklara hur dessa lösningar fungerar, är du godkänd i laborationsmomentet. Alla program måste vara skrivna på ett snyggt och begripligt sätt. Programraderna skall t.ex. indenteras (dras in) så att det underlättar läsbarhet och förståelse. Du måste också ha delat in det i "moduler" med användning av include-filer så som beskrivs i detta lab-pm.

Uppgift 1

Skapa ett nytt workspace *lab4.w12*. I detta workspace skall du sedan skapa ett nytt projekt för varje uppgift i laborationen.

Lägg in ett nytt projekt *uppgift1.m12*.

En laborationsmodul ML4 skall anslutas till mikrodatorsystemet via en inport och en utport. På ML4 finns bl.a. en s.k. *DIP-switch,* som är en enhet med åtta små switchar. Laborationsdatorn kan avläsa switcharnas lägen via inporten ML4IN, som har adressen 0x600. Porten har åtta bitar och varje bit motsvarar läget för en switch. ML4 har också en s.k. *Parallell output,* som är en enhet med åtta lysdioder. Man kan tända och släcka lysdioderna genom att skriva till utporten ML4OUT, som har adressen 0x400.

Uppgiften är att skriva ett program som avläser switcharnas lägen på ML4 och som visar de avlästa lägena genom att skriva till *Parallell output*-enheten.

Programmet skall utformas som en "Stand alone" applikation. (Se sid 73 i ARB1). Detta betyder att du inte får använda *Standard startup* när du anger egenskaperna för ditt program i dialogrutan *Settings*. Du skall istället skriva en egen startsekvens i assembler. Denna sekvens skall anropa din `main`-funktion.

Läsningen och skrivningen av ML4-portarna skall ske i `main`. Du skall skapa en speciell fil med namnet `ports.h`. (Jämför "*Absolut adressering i C*" sid 74 i ARB1). Denna skall innehålla definitioner av makron som beskriver portadresserna. Inga andra portadresser får användas i funktionen `main`. Ditt projekt skall alltså bestå av tre filer, filen `ports.h`, en assemblerfil med startsekvensen samt en C-fil med funktionen `main`. Skapa dessa filer och lägg dem till projektet.

Innan du kompilerar ditt program skall du kontrollera att konfigurationen är satt till *Debug*. (Man bestämmer konfiguration med menyalternativet *Build* | *Configuration*. Om *Debug* är gråmarkerat är konfigurationen redan satt till *Debug*.) Testa din lösning i källkodssimulatorn i XCC. För att kunna göra detta måste du ansluta en ML4 *Dip-switch* och en ML4 *Parallell output*-enhet med hjälp av knappen *Simulator setup* på simulatorns verktygslist. Du skall inte ansluta något *Console*- fönster.

När denna uppgift fungerar har du en startsekvens som kan återanvändas i de följande uppgifterna i laborationen. Du har också lärt dig hur man skapar projekt och hur man kör och konfigurerar simulatorn. Detta kommer du att ha nytta av i fortsättningen.

VID LABORATIONSTILLFÄLLET:

Följande ska visas för handledaren:

- Kompilera projektet.
- Öppna en terminal mot laborationsdatorn. Kontrollera att MC12 och ML4 kopplats samman korrekt.
- Ladda ner programmet och starta det med "trace". Stega fram till den första instruktionen i "main".
- Starta därefter programmet med "go" och kontrollera funktionen.

Uppgift 2

Skapa nu ett nytt projekt i samma workspace som tidigare. Kalla det nya projektet *uppgift2.m12* och gör det till det aktiva projektet. Du kan börja med att lägga din assemblerfil med startsekvensen till det nya projektet. Resten av programkoden i projektet skall skrivas i C.

I denna uppgift ska en funktion konstrueras:

```
unsigned char  ML15_Keyboard( void );
```

`ML15_Keyboard` läser från tangentbordet ML2. Tangentbordet har 16 knappar numrerade från 0 till 15. När funktionen körs i ett program skall den returnera en knappkod när användaren trycker på någon av knapparna. Funktionen skall först vänta tills alla knapparna är uppe. Därefter skall koden för den först nedtryckta knappen bestämmas och returneras. Koden som avläses från tangentbordet framgår av figur 3 i ”*Arbetshäfte för laboration nr 1-3*” medan koden som skall returneras visas i figur 4 eller i Appendix 7, Tangentbordsdisposition, sidan 45, i samma häfte.

På laborationsmodulen ML15 finns inporten *Key Decode Register* (adress 0x9C0). Beskrivning av bitarna på denna port finns i hjälptexten till XCC.

Filerna `keyboardML15.c` och `keyboardML15.h`. skall innehålla funktionen `ML15_Keyboard` respektive prototypdeklarationer. Definitioner av makron för portadresserna skall läggas i filen `ports.h` från uppgift 1.

Skriv ett huvudprogram som läser av tangentbordet och visar resultatet på *Parallell output* på ML4. Lägg till programfilen till projektet och kompilera.

Testa ditt program i simulatorn. Då du ansluter modulen ML2 till simulatorn skall du vara noga med att välja *Interface* | *ML15*. När du klarat av detta steg har du en funktion för att läsa indata. Du kommer att behöva funktionen för att kunna lösa den sista uppgiften.

VID LABORATIONSTILLFÄLLET:

Följande ska visas för handledaren:

- Kompilera projektet.
- Öppna en terminal mot laborationsdatorn. Kontrollera att MC12, ML15 och ML23 kopplats samman korrekt.
- Ladda ner programmet och starta det med ”go” och kontrollera funktionen.

Uppgift 3

Skapa ytterligare ett nytt projekt i samma workspace som tidigare. Kalla det nya projektet *uppgift3.m12* och gör det till det aktiva projektet. Även denna gång kan du börja med att lägga din assemblerfil med startsekvensen till det nya projektet.

I denna uppgift skall du skriva ett program som hanterar borrmaskinen du stiftade bekantskap med under laborationerna 1-3. Det program du skall skriva nu skall i stort göra samma sak som det assemblerprogram du skrev då, men alla program skall nu skrivas i C. Funktionerna som styr borrmaskinen skall samlas i en källtextfil och en header-fil med namnen `drill.c` och `drill.h`.

Följande funktioner skall ingå:

- `init`
- `start`
- `stop`
- `down`
- `up`
- `step`
- `ddtest`
- `drill`
- `refpo`
- `auto_drill`
- `alarm`
- `nstep`
- `delay`
- `outone`
- `outzero`

Dessa funktioner skall utföra exakt samma saker som motsvarande assembler-funktioner i laboration 1-3, men de skall vara skrivna i C. Alla funktionerna, utom de fem sista, är parameterlösa och saknar returvärde.

Funktionerna `alarm, nstep, delay, outone` och `outzero` skall ha ett 8 bitars heltal utan tecken som parameter. De ger inget resultatvärde.

Börja med att göra en flödesplan för hela programmet. Flödesplanen ska visas upp vid laborationstillfället. När du konstruerar funktionerna för borrmaskinen behöver du ibland lägga in fördröjningar. Precis som i tidigare laborationer kan dessa komma att behöva utformas på något olika sätt beroende på om du testar ditt program i simulatorn eller i hårdvara. Nu skall borrmaskinen kopplas in igen, på samma sätt som i laboration 1-3. Den skall styras via utporten som har adressen 0x400. Naturligtvis skall alla makrodefinitioner läggas i filen `ports.h`. Kom ihåg att utporten inte är läsbar så du måste använda ett ”skuggregister”, en kopia av det senaste värdet som skrevs till porten.

Ditt program skall läsa från tangentbordet och om någon av tangenterna 0-6 trycktes ner skall en av funktionerna `start`, `stop`, `down`, `up`, `step`, `drill` resp `auto_drill` anropas. Om någon annan tangent tryckts ner skall inget utföras. (Här är det lämpligt att använda en **`switch`**-sats.) För att läsa från tangentbordet använder du förstås inläsningsmodulen från uppgift 2.

VID LABORATIONSTILLFÄLLET:

Följande ska visas för handledaren:

- Flödesplan för programmet.
- Kompilera projektet.
- Öppna en terminal mot laborationsdatorn. Kontrollera att MC12, ML15, ML23 och borrmaskinen kopplats samman korrekt.
- Ladda ner programmet och starta det med ”go”. Kontrollera funktionen.

Uppgift 4

Skapa nu projektet *uppgift4.m12* i samma workspace som tidigare och gör det till det aktiva projektet.

I denna uppgift skall du komplettera uppgift 3 med de händelsestyrda avbrotten för ”*Signal*” och ”*Nödstopp*” med samma hårdvara och beteende som i laboration 1-3. Pseudoparallellism (sysini) skall inte användas i laboration 4.