Laboration 5: Ett komplett system - Tärningsspel

1 Inledning

Laborationen ska bidra till en grundläggande förståelse för:

- Design och implementation av ett program skrivet med assembly-kod
- Struktur (i filer) av ett program skrivet med assembly-kod.
- Access till olika typer av data som finns i programminnet och dataminnet.

1.1 Utrustning

Komponenter som behövs för denna laboration: Ingen ny hårdvara behövs

2 Beskrivning av laboration

I den här laborationen ska ni sätta ihop delsystem som skapats i de tidigare labbarna (subrutiner för avläsning av tangentbord och för att skriva till LCD) och sätta ihop det på ett snyggt sätt till ett fungerande system, med god struktur och dokumentation. För att klara av laborationen måste ni, precis som i tidigare laborationer, förbereda laborationen genom att läsa instruktionerna noggrant, studera teori och datablad, etc. Laborationen består av följande moment:

1. Strukturering och dokumentering av programkod
2. Programmering av tärningsapplikation
3. Lagring av statistikdata
4. Test av komplett system

2.1 Syfte och mål

Laborationen går ut på att sätta ihop delsystem som skapats i tidigare labbar till ett komplett system. Systemet skall ”snyggas till” så att det har bra struktur och är väl kommenterat.

2.2 Examination

2.2.1 Inlämning av rapport och programkod

Utöver en rapport ska ni även lämna in programkod. Instruktioner för detta är beskrivet på inlämningssidan för denna laboration (på It’s Learning).

2.2.2 Praktisk och muntlig redovisning

Den praktiska delen av laborationen examineras efter att laborationens sista moment har avslutats. Följande ska redovisas:

- Programkod (med god struktur och kommentarer).
- Demonstration av systemet, fullt fungerande enligt specifikation.

Ni ska även kunna redogöra för funktionalitet avseende krets och programkod.

3 Moment 1: Strukturering och dokumentering av programkod.

3.1 Beskrivning

I detta moment ska ni använda rutiner från tidigare lab (Lab 4) och strukturera upp denna kod innan ni går vidare.

3.2 Konfigurering av projekt och strukturering av kod

Uppgift 3.2.1

Skapa ett nytt assembler-projekt i Atmel Studio (processorn är ATMega2560). Kalla det för **lab5**.

Uppgift 3.2.2 (redovisas i programkod)

Kopiera och inkludera alla kod-filer ni hade i lab 4 (**xxxx.inc)** i ert projekt. Kom även ihåg att anpassa huvudprogrammet för detta!

Uppgift 3.2.3 (redovisas i programkod) Passa på att byta ut NOP-instruktionerna i tangentbordsrutinen mot anrop till lämplig delay-funktion (ca 2st anrop till delay_1_micros kan vara lagom delay för att rutinen skall fungera och samtidigt hantera knappstuds).

Har ni inte redan gjort en subrutin: **delay_1_s** bland era delayrutiner, så behöver ni göra detta nu (annars får ni fel då ni inkluderar de färdiga rutinerna stat.inc och monitor.inc).

Uppgift 3.2.4 (redovisas i programkod)

Se till att varje fil (huvudprogrammet och alla include-filer) är strukturerade, (tex) efter följande mall:

1) Kommentarblock med:
    a. Beskrivning av innehållet i filen
    b. Namn på de som skapat/editerat filen
    c. Datum
2) Definitioner
3) Konstanter
4) Init-rutiner
5) Interna rutiner (som inte är tänkta att anropas ”utifrån”)
6) Externa rutiner (som skall kunna anropas ”utifrån”)
7) Main-program (i förekommande fall)

Uppgift 3.2.5 (redovisas i programkod)

Se till att all kod är väl kommenterad, dvs beskriver vad som logiskt händer, INTE vad assembler-instruktionen innebär! Subrutiner skall ha några rader före rutinen, som innehåller:

- Vad subrutinen gör
- Hur in/utparametrar hanteras (tex vilka register som används)
- Vilka register som används internt i rutinen (och som alltså efter rutinen exekverats inte kommer att innehålla det de innehöll innan anropet)
- Vilka begränsningar som finns (tex max-storlekar, etc.) och ev felhantering

```
;-------------------------------------------------------------
: Example_Subroutine. Converts a string to <something>
; Parameters IN:    R24: contains number.
;                   R16: contains position (0..16).
;            OUT:   R24: contains ASCII character.
;   Limitations: If R16 <0 or R16 >16, no action is taken.
;-------------------------------------------------------------
```

Uppgift 3.2.6 (redovisas i programkod)
Se till att huvudprogrammet inte innehåller ”för mycket kod”. Huvudprogrammet skall vara ganska litet och innehålla:

- Anrop till subrutiner för att initiera systemet
- En oändlig loop, innehållande tex:
    - Kod för enkla tester (kommenteras ut efter att testerna körts, tas bort i slutversion)
    - Anrop till subrutiner. I denna lab är det lämpligt att implementera huvud-loopen i figur 6-1 i huvudprogrammet.

Om huvudprogrammet innehåller för mycket kod, gör nya subrutiner och flytta delar dit.

Uppgift 3.2.7
Bygg systemet, ladda ner och verifiera att programmet fortfarande fungerar som tidigare.

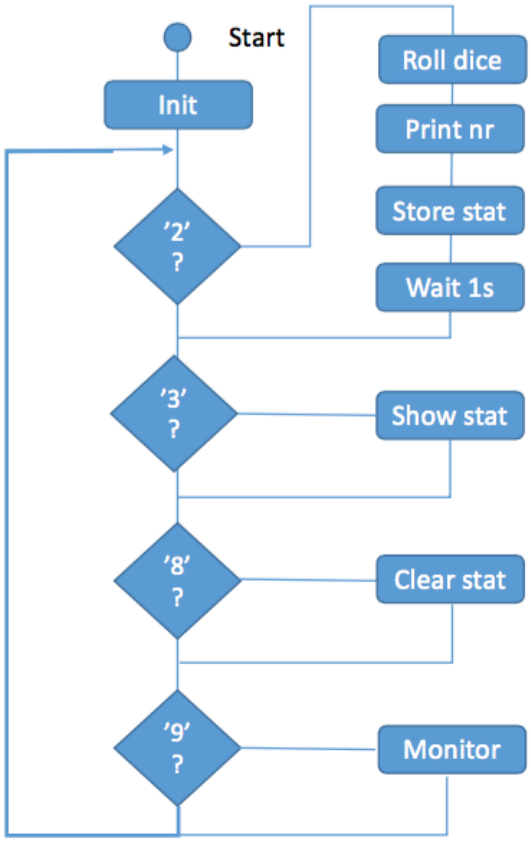
4 Moment 2: Programmering av tärningsapplikation

4.1 Programmering av applikation

Skapa en ny include-fil (tex Tarning.inc) och skriv programmet för tärnings-applikationen, baserat på tärningsexemplet från Föreläsning 10. Lägg till lämpliga anrop från huvudprogrammet. Ni bör även lägga till andra saker, tex en välkomsttext och hantering av fler knappar, men funktionaliteten i aktivitetsdiagrammet nedan (den del som berör hantering av knapp ’2’) skall finnas med.

Texterna kan vara tex: ”WELCOME! ” (tänk på att alla strängar bör innehålla ett jämnt antal tecken – inklusive 0’an - annars ger assemblern en varning…), vänta 1s och skriv sedan tex: ”PRSS ’2’ TO ROLL”. När man trycker på 2’an bör ni skriva ut: ”ROLLING…”. Efter att man släppt tangenten skriver ni tex ”VALUE:” och tärningens värde.

Lagra strängarna i programminnet och använd den MACRO ni definierade i Lab 4 för att och skriva ut dem.



Figur 4-1 Principiellt Aktivitetsdiagram för Tärningsprogram (ej komplett!)

Uppgift 4.1.1 (redovisas i programkod)
Skriv subrutinerna som behövs för kärnfunktionaliteten i Tärningsapplikationen (dvs anrop till subrutiner för avkänning av tangent och utskrift på LCD. Vänta med strängarna. Bygg och testa att det fungerar.

Uppgift 4.1.2 (redovisas i programkod)
Inkludera filerna `stats.inc` och `monitor.inc` på lämpligt ställe i huvudprogrammet. Filerna hämtas från It's learning. (Ni behöver kommentera bort anrop till rutiner som inte finns än). Glöm inte att anropa `init_stat` och `init_monitor`!

Utöka koden så att strängar som behövs skrivs ut. Följ aktivitetsdiagrammet, men lägg till. Notera att rutinerna ”showstat”, ”clearstat” och ”monitor” innehåller egna strängar som skrivs ut. Bygg och testa att det fungerar.

5 Moment 3: Insamling och lagring av statistik

5.1 Beskrivning

I detta steg skall ni lägga till en datastruktur och subrutiner för att lagra och hämta ut statistik (som visar om tärningen är rättvis).

Uppgift 5.1.1 (redovisas i programkod)
Deklarera ett antal variabler i dataminnet. Följande värden kan vara intressanta:

- Plats för en räknare som visar antal kast.
- Plats för en räknare som visar antalet gånger siffran ’1’ har kommit upp.
- Plats för en räknare som visar antalet gånger siffran ’2’ har kommit upp.
- Plats för en räknare som visar antalet gånger siffran ’3’ har kommit upp.
- Plats för en räknare som visar antalet gånger siffran ’4’ har kommit upp.
- Plats för en räknare som visar antalet gånger siffran ’5’ har kommit upp.
- Plats för en räknare som visar antalet gånger siffran ’6’ har kommit upp.

Skapa en fil (tex **stat_data.inc**) som tex innehåller följande:

```
/* ------------------------------------------------

Space in the RAM to save the results from dice throws.

The following functions are provided:

            store_stat (R24 contains the dice value)

                        The function will increment the

                        Total number of throws and the

                        number of throws with results equals R24.

            get_stat (R24 gives the value for which the

                        result is retrieved. If R24 = 0, the

                        total number of throws shall be returned.

                        The result is returned in R24.

            clear_stat (no parameters in nor out)

                        Clears all data.

------------------------------------------------------*/

            .DSEG                   ; The following applies to the RAM:

            .ORG        0x200       ; Set starting point
                                    ; address of data
                                    ; segment to 0x200

<your_label>:           .BYTE       <n>         <skapa så mycket plats som behövs.
Antag att max värde per lagrad variabel är 255>

.CSEG

store_stat: <skriv kod för rutinen>

                        RET

get_stat:   <skriv kod för rutinen>

                        RET

clear_stat: <skriv kod för rutinen>

                        RET
```

Skriv sedan kod i huvudprogrammet så att `store_stat` anropas då programmet körs.

Tips: En array med 7 platser kan användas och värdet på ”kastet” kan användas som index för att komma åt ”rätt” räknare.

Uppgift 5.1.2 (redovisas i programkod)
Ändra nu i huvudprogrammet, så att alla delar av aktivitetsdiagrammet finns med, dvs man skall kunna trycka på olika knappar för att rulla tärningen, se resultat av statistik, rensa statistik och köra monitorn.

6 Moment 4: Test av komplett system, redovisning och reflektion

6.1 Beskrivning

I detta, sista steg skall ni kolla att programkoden fortfarande är väl strukturerad och att allt fungerar som det skall.

Uppgift 6.1.1 (redovisas i rapport)
Testkör programmet! Om det fungerar som det ska är det dags att undersöka om tärningen ger rättvisa värden:

- Tryck ’8’ (rensa resultat)
- Rulla tärningen ca 100 gånger (Håll inne 2’an så att ”Rolling…” hinner skrivas ut och dice-rutinen hinner köra en stund.
- Tryck ’3’ för att visa resultaten och skriv ner dessa i en tabell. Räkna även ut procentsatsen för varje värde, enligt exempel nedan. **Notera att siffrorna skrivs ut som hextal!**

Antal kast:	117 (0x75)	procent
Antal ’1’	24 (0x18)	20%
Antal ’2’	24 (0x18)	20%
Antal ’3’	16 (0x10)	14%
Antal ’4’	16 (0x10)	14%
Antal ’5’	15 (0x0F)	13%
Antal ’6’	22 (0x16)	19%

Uppgift 6.1.2 (redovisas i programkod)
Nu är det dags att avsluta genom att:

- Se till att ni förstår vad programmet gör i alla delar.
- Snygga till koden och kommentera programmet där det saknas kommentarer. Ni ska kunna återvända till koden om några månader och fortfarande förstå vad som sker!
- I kommentarsblocket i början av filerna ska ni skriva in era namn och aktuellt datum. Dessutom ska ni med egna ord beskriva vad programmet gör.

När detta är klart kan ni redovisa för labhandledare!

Uppgift 6.1.3 (redovisas i rapport)
Redogör för era erfarenheter från denna laboration. Vad har ni lärt er? Gick allting bra eller stötte ni

på problem? Om allting gick bra, vad var i så fall anledningen detta? Om ni stötte på problem, hur löste ni i så fall dem?

Fördjupande uppgifter (görs om tid finns)

7 Moment 5: Förbättring av statistikpresentation och monitor

7.1 Beskrivning

I detta, extra steg skall ni granska koden för att visa statistiken och monitorn. Ni kan hitta på andra sätt att lösa uppgifterna (se nedan vad programmen gör) och sedan implementera/testa detta.

Uppgift 7.1.1 (redovisas i programkod)
Statistikpresentationen visar 7 värden i sekvens enligt följande:

- Skriv ut: ”THROWS:” och visar totalt antal kast (dvs anrop till `get_stat` med R24 = 0)
- Vänta 1 s
- Skriv ut: ”NBR OF 1’S:” och visar antal 1’or (dvs anrop till `get_stat` med R24 = 1)
- Vänta 1 s
    - *<samma sak för siffrorna 2-6>*
- Hoppa tillbaka till huvudprogrammet

Förbättra koden. Testa!

Uppgift 7.1.2 (redovisas i programkod)
Statistikrensningen visar texten ”CLEARING…” och anropar sedan `clear_stat`. Sedan hoppa tillbaka till huvudprogrammet.

Förbättra koden. Testa!

Uppgift 7.1.3 (redovisas i programkod)
Monitor-programmet är ett litet hjälpprogram, som tillåter att man läser av värden i dataminnet. Då programmet startar, skrivs”>0000:xx” ut (där ”xx” är 2 hex-siffror för innehållet i cellen.

- Om man trycker på ’#’, visas innehållet i nästa minnescell. (Om man håller inne knappen, så räknar programmet upp räknaren och visar samtidigt innehållet i cellerna…
- Om man trycker på knapparna ’0’ – ’9’, så byts startadressen till motsvarande startaddress, där den höga adressen blir värdet på tangenten som trycks ner, tex >0200:xx då 2’an trycks ner. (på adress 0200 skall ju era data finas….. Man kan på adress 0000 och framåt se innehållet i alla register!
- Om man trycker på ’*’, avbryts monitorn och man hoppar tillbaka till huvudprogrammet.

Förbättra koden. Testa!