Laborversuch 1

Thema:

Einführung in die HCS12 Assemblerprogrammierung

Inhaltsverzeichnis

Zusätzlich erforderliche Unterlagen:

- Vorlesungsmanuskript zu Kapitel 1 und 2 mit Anhang Metroworks CodeWarrior HCS12 Entwicklungsumgebung
- Dokumentationspaket für den HCS12 Mikrocontroller und das Dragon12-Entwicklungsboard

1. Überblick

Ziel des Laborversuchs

- Kennenlernen der Metroworks CodeWarrior HCS12 Entwicklungsumgebung
- Programmieren und Testen einfacher HCS12 Assemblerprogramme

Ablauf des Laborversuchs

Der Laborversuch besteht aus einem Vorbereitungsteil und einem Teil, der erst im Labor durchgeführt wird. Der Vorbereitungsteil enthält eine Reihe von Aufgaben, die **vor dem Laborversuch** bearbeitet werden müssen und in diesem Umdruck mit **Vorbereitungsaufgabe** gekennzeichnet sind. Bei der Vorbereitung werden Assemblerprogramme erstellt und, soweit möglich, mit dem Simulator getestet. Im Labor werden diese und weitere Programme auf das Dragon12-Entwicklungsboard portiert und auf der realen Hardware erprobt.

(1) **Notwendige Vorkenntnisse und Vorbereitungsaufwand**: Für die Vorbereitung dieses Laborversuchs muss ausreichend Zeit eingeplant werden. Am Ende der Vorbereitung sollten Sie soviel Übung im Umgang mit der CodeWarrior-Entwicklungsumgebung haben, dass Sie HCS12-Programme selbständig erstellen können, ohne ständig in den Unterlagen oder der Online-Hilfe nachsuchen zu müssen.

(2) **Laborunterlagen: Den vollständigen Laborumdruck sowie vorbereitete CodeWarrior-Projekt-Dateien finden Sie auf der Web-Seite zu dieser Vorlesung**. Entwerfen Sie alle Programme zunächst mit Hilfe eines Programmablaufplans (Flußdiagramm), bevor Sie den Programmcode schreiben. Vergessen Sie nicht, Ihre Programme ausführlich zu kommentieren. Bringen Sie zum Laborversuch sämtliche Vorbereitungsunterlagen inklusive der Programmablaufpläne und der CodeWarrior-Projektdateien mit.

Die Vorbereitung sollte in der Laborgruppe gemeinsam so durchgeführt werden, dass jeder Gruppenteilnehmer in der Lage ist, den Laborbetreuern die Lösung zu sämtlichen Vorbereitungsaufgaben selbständig zu erklären und bei Bedarf sämtliche Programme vorzuführen und zu erläutern.

Bitte führen Sie die Vorbereitung ohne Rückgriff auf Altmeister durch, auch wenn das eine deutliche Zeitersparnis bedeuten würde. In der Klausur werden Sie auch keinen Altmeister haben und die Labors sind in dieser Lehrveranstaltung die allerbeste Prüfungsvorbereitung. Softwareentwicklung in einer beliebigen Programmiersprache kann man nur dann wirklich lernen, wenn man es selbst macht!

Mangelhafte Vorbereitung oder unentschuldigtes Fehlen führt zum Nichtbestehen des Labors. Das nichtbestandene Labor kann dann im folgenden Semester wiederholt werden.

(3) **Labordurchführung:** Während des Laborversuchs benötigen Sie in jedem Fall das Vorlesungsmanuskript sowie die vollständige Dokumentation des HCS12-Mikrocontrollers und seiner Peripheriebausteine sowie des Dragon12-Entwicklungsboards. Bringen Sie diese Unterlagen bitte zum Labor mit. Während des Labors sind die als **Laboraufgabe** gekennzeichneten Fragen zu bearbeiten.

2. Einführung in die CodeWarrior HCS12-Entwicklungsumgebung

Vorbereitungsaufgabe 2.1

Studieren Sie den Anhang „Metroworks CodeWarrior HCS12 Entwicklungsumgebung“ des Vorlesungsmanuskriptes. Installieren Sie die CodeWarrior-Entwicklungsumgebung gegebenenfalls auf Ihrem eigenen Rechner.

Machen Sie sich anhand der Programme BlinkingLeds und BlinkingLedsAsm, die in der Vorlesung in Kapitel 2.2 behandelt wurden, mit der Programmierung des HCS12 sowie dem Übersetzen und Debuggen von Programmen in der CodeWarrior-Entwicklungsumgebung vertraut. Probieren Sie insbesondere aus, wie man Progammänderungen durchführt, den Simulator startet und im Simulator im Einzelschrittbetrieb und mit Hilfe von Haltepunkten den Programmablauf untersuchen kann.

Seien Sie darauf vorbereitet, im Labor diesbezügliche Fragen beantworten zu können.

Vorbereitungsaufgabe 2.2

Erstellen Sie basierend auf der Vorlage „**lab1a**“ ein kleines Assemblerprogramm, das zyklisch einen 8bit-Wert hochzählt und in binärer Form auf der LED-Zeile des Dragon12-Entwicklungsboards ausgibt. Das Programm soll eine Warteschleife enthalten, so dass das Hochzählen annäherend im Sekundentakt erfolgt. Die Warteschleife sollte als Unterprogramm implementiert werden.

Demonstrieren Sie die Funktionsfähigkeit Ihres Programms im Simulator der Entwicklungsumgebung. Benutzen Sie zur optisch ansprechenden Darstellung die „IO_Led“-Komponente des Simulators. Justieren Sie die Warteschleife mit dem Simulator so, dass sich der gewünschte Sekundentakt ergibt. Gehen Sie davon aus, dass die CPU mit einem Takt (BUSCLK) von 24MHz arbeitet.

Vorbereitungsaufgabe 2.3

Erstellen Sie basierend auf der Vorlage „**lab1b**“ ein Assemblerprogramm, das den Mittelwert der einzelnen Elemente des vorgegebenen Zahlenvektors **`vektor`** bestimmt und den Wert in die Variable **`ergebnis`** schreibt. Das Programm soll so arbeiten, dass man keinen Programmcode ändern muss, wenn sich die Vektorlänge ändert. Sie dürfen vereinfachend davon ausgehen, dass es bei der Berechnung des Mittelwertes keine Überläufe gibt, wenn Sie mit 16bit Zwischenwerten arbeiten.

Entwerfen Sie das Programm zunächst mit Hilfe eines Programmablaufplans und schreiben Sie erst dann den Programmcode.

Demonstrieren Sie im Simulator, dass Ihr Programm richtig arbeitet. Zeigen Sie auch, dass Sie die Anzahl der Vektorelemente vergrößern und verkleinern können, ohne dass Sie den eigentlichen Programmcode ändern müssen.

Benutzen Sie die Linker-Map-Datei um herauszufinden, bei welcher Speicheradresse das Ergebnis des Programms abgelegt wird.

Vorbereitungsaufgabe 2.4

Erstellen Sie basierend auf der Vorlage „**lab1c**“ ein Unterprogramm **`toLower`** in Assembler, das einen Null-terminierten String in Kleinbuchstaben konvertiert. Der String soll sowohl Großbuchstaben als auch Kleinbuchstaben, aber auch Zahlen und Sonderzeichen enthalten dürfen.

Für die Konvertierung zwischen Klein- und Großbuchstaben kann die Tatsache verwendet werden, dass die Kodierung des ASCII-Codes so gewählt ist, dass gilt

$$\text{ASCII-Code}_{\text{Kleinbuchstabe}} = \text{ASCII-Code}_{\text{Großbuchstabe}} + 32_D$$

Das Unterprogramm erhält beim Aufruf im D-Register einen Zeiger auf den String, der String selbst stehe in einem Array im Speicher. Aus Sicht des aufrufenden Programms soll das Unterprogramm den Inhalt der Register D, X und Y nicht verändern.

Das Unterprogramm soll in einer eigenen Datei „toLower.asm“ stehen. Schreiben Sie dazu in „main.asm“ ein Testprogramm für das Unterprogramm **`toLower`**. Entwerfen Sie das Programm zunächst mit Hilfe eines Programmablaufplans, bevor Sie mit dem Programmieren beginnen.

Demonstrieren Sie im Simulator, dass Ihr Programm richtig arbeitet.

Vorbereitungsaufgabe 2.5

Erstellen Sie ein Unterprogramm **`hexToASCII`** in Assembler, das einen 8-Bit hexadezimalen Wert in einen ASCII-String konvertiert. Beispiel: Der Wert sei „$F8“. Zurückgegeben werden sollen dann die ASCII-Werte von „F“ und „8“, also 46_H und 38_H. Der Wert soll als Byte im Register B an das Unterprogramm übergeben werden. Der Rückgabewert soll im Register D stehen. Aus Sicht des aufrufenden Programms soll das Unterprogramm den Inhalt der Register X und Y nicht verändern.

Die Umwandlung einer einzelnen Hexadezimalstelle (4bit) in den ASCII-Code kann z.B. mit Hilfe einer Tabelle vorgenommen werden, die im ROM folgendermassen definiert

```
H2A:  DC.B „0123456789ABCDEF“
```

und über den Hexadezimalwert als Index ausgelesen wird.

Das Unterprogramm soll in einer eigenen Datei „hexToASCII.asm“ stehen. Schreiben Sie dazu in „main.asm“ ein Testprogramm für Ihr Unterprogramm **`hexToASCII`**. Entwerfen Sie das Programm zunächst mit Hilfe eines Programmablaufplans, bevor Sie mit dem Programmieren beginnen.

Schreiben Sie ein Hauptprogramm, bei in einer Schleife ein 8bit-Zähler inkrementiert und das Unterprogramm **`hexToASCII`** mit dem Zählerwert als Parameter aufgerufen wird und demonstrieren Sie im Simulator, dass Ihr Programm bei verschiedenen Zählerwerten korrekt arbeitet.

Vorbereitungsaufgabe 2.6

Erstellen Sie ein Unterprogramm **`decToASCII`** in Assembler, das eine 16-Bit vorzeichenbehaftetete Zahl in einen ASCII-String konvertiert, der dem Dezimalwert der Zahl entspricht. Der Wert soll als Wort im Register D an das Unterprogramm übergeben werden. Im X-Register soll ein Zeiger auf den Ort übergeben werden, an dem der Null-terminierte ASCII-String gespeichert werden soll. Vergessen Sie nicht, ausreichend Speicherplatz zur Verfügung zu stellen (mit Vorzeichen und dem abschließenden 0_H-Byte insgesamt 7 Byte). Bei negativen Werten soll ein „-"-Zeichen vorangestellt werden, bei positiven Werten soll vor der Zahl ein Leerzeichen erscheinen. Innerhalb des Feldes sollte die Zahl rechtsbündig stehen. Aus Sicht des aufrufenden Programms soll das Unterprogramm den Inhalt der Register D, X und Y nicht verändern.

Das Unterprogramm soll in einer eigenen Datei „decToASCII.asm" stehen. Schreiben Sie dazu in „main.asm" ein Testprogramm für die Funktion **`decToASCII`**. Entwerfen Sie das Programm zunächst mit Hilfe eines Programmablaufplans, bevor Sie mit dem Programmieren beginnen.

Demonstrieren Sie im Simulator, dass Ihr Programm richtig arbeitet. Testen Sie dabei insbesondere die Werte +32767, -32768, +9001, +255, +256, -256, 100, -99 und 0. Sie sollen das Unterprogramm dabei in Ihrem Testprogramm mehrfach hintereinander mit den unterschiedlichen Werten aufrufen, um zu überprüfen, ob das Unterprogramm auch bei mehreren aufeinanderfolgenden Aufrufen korrekt arbeitet.

Testen Sie die beiden Unterprogramme `hexToASCII` und `decToASCII` bitte sorgfältig! Beide Programme werden in Teil 3 und in den folgenden Labors eingesetzt. Für eine umfangreiche Fehlersuche bleibt im Labor keine Zeit!

3. Programmtests auf dem Dragon12-Entwicklungsboard

Laboraufgabe 3.1

Portieren Sie das in der Vorbereitungsaufgabe 2.2 erstellte Programm auf die Zielhardware und demonstrieren Sie seine Funktionsfähigkeit. Überprüfen Sie, ob das Hochzählen tatsächlich im Sekundentakt erfolgt.

Ändern Sie das Programm so ab, dass es nach kurzer Betätigung des Tasters SW5 die Zählrichtung umkehrt. Verwenden Sie dabei keinen Interrupt, sondern „Polling", d.h. Abfragen des Tasters in der Programmschleife.

Laboraufgabe 3.2

Sie erhalten im Labor eine Vorlage, die eine Ausgabe einer Null-terminierten Zeichenkette auf das LCD erlaubt. Leider haben sich dort zwei kleine Fehler eingeschlichen. Reparieren Sie das Programm, so dass es ordnungsgemäß funktioniert. Dazu werden sie eventuell die Dokumentation des LCD-Controllers studieren müssen. Sie finden diese im Dokumentationspaket zur Vorlesung.

Laboraufgabe 3.3

Fügen Sie die in Vorbereitungsaufgabe 2.5 und Vorbereitungsaufgabe 2.6 erstellten Programme mit dem aus Laboraufgabe 3.2 so zusammen, dass Ihr Testprogramm in einer Schleife einen 16bit Wert i inkrementiert. Der dezimale Wert von i soll in Zeile 1 des LCD ausgegeben werden, der hexadezimale Wert in Zeile 2. Dabei soll vor dem hexadezimalen Wert eine führende „0x" angezeigt werden.

Die Zählschleife soll so langsam ablaufen, dass das Hochzählen ungefähr im Sekundentakt erfolgt.

Erweitern Sie die Zählschleife so, dass der Zähler rückwärts statt vorwärts zählt, während die Taste SW5 auf dem Dragon12-Entwicklungsboard gedrückt wird. Wenn man die Taste loslässt, soll der Zähler wieder vorwärts zählen.