Fakultät für Ingenieurwissenschaften
Labor für Eingebettete Systeme

ingenieur
wissenschaften
htw saar

Studiengang
Praktische Informatik

Praktikum
Mikroprozessortechnik

Erstellt von Dipl.-Ing. H.-J. Bohr
Stand Wintersemester 2024/2025

Inhalt:

1. Allgemeines zur Versuchsdurchführung

Im ersten Teil des Praktikums „Mikroprozessortechnik“ werden die Aufgaben zur Programmierung in Assembler nur mit dem PC, der Demoversion der Entwicklungsumgebung „µVision“ von Keil und dem zugehörigen Simulator durchgeführt. Im zweiten Teil, bei der Programmierung der Peripheriekomponenten (Programmiersprache C), wird zusätzlich der Experimentiercomputer ARM-MP als Zielsystem eingesetzt.

Der Experimentiercomputer ist mit dem Mikrocontroller LPC 2194 der Firma NXP (ehemals Philips) aufgebaut, der einen ARM7-Prozessorkern beinhaltet. Als Lern- und Übungsgerät erlaubt der Experimentiercomputer das Testen und Ausführen von Programmen.

Während eines Termins ist immer ein Laborversuch zu bearbeiten. Die Übungen sind, wie in den Aufgaben beschrieben, durchzuführen. Abhängig von den Vorkenntnissen wird wahrscheinlich eben so viel Zeit für die Vorbereitung der Laborversuche benötigt wie für ihre praktische Ausführung. Die erarbeiteten Lösungen sind vorzuführen.
Nach der Bearbeitung der Laborversuche sollen grundlegende Kenntnisse und Erfahrungen über Mikrocontroller und deren Peripheriekomponenten, sowie die Programmerstellung in Assembler und der Programmiersprache C vorhanden sein.

2. Laborversuch 1: Assemblerprogrammierung

Hinweise:

- Zur Bearbeitung der Aufgaben zur Assemblerprogrammierung kann das bereitgestellte Projekt
- „Assembler_Uebung“ verwendet werden.
- Achten Sie bei der Programmierung auf die korrekte Parameterübergabe bei den Unterprogrammaufrufen.
- Verwenden Sie nur die Speicherbereiche, die Ihnen in den Aufgabenstellungen zugewiesen wurden.
- Zum Zwischenspeichern von Daten können, sofern nichts anderes angegeben ist, die Register und der Stack verwendet werden.
- Verwenden Sie nur den ARM-Befehlssatz (keine Thumb-Befehle).
- Definieren Sie Konstanten oder Ausdrücke mit der Pseudo-Anweisung bzw. Directive EQU als aussagekräftige, selbsterläuternde Symbole am Anfang des Moduls, vor den Unterprogrammen oder in einer separaten Include-Datei.
- Die Multiplikation einer Zahl mit einer Konstanten sollte, wenn möglich, durch die Berechnung mittels Zweierpotenzen realisiert werden (s. Skript).
- Ebenso sollte die Division durch eine Konstante entweder durch die Berechnung mittels Zweierpotenzen oder durch die Multiplikation des inversen Elements realisiert werden (s. Skript).
- Die nachfolgenden Aufgaben sind alle zu bearbeiten und auszutesten.
- Die Aufgaben sind vorzuführen. Von allen Aufgaben ist ein Ausdruck der Programmquelle (mit Namen und Matrikel-Nr.) abzugeben.

2.1 Funktion AtouI (= Ascii to unsigned Integer)

Im Datenspeicher steht ein String (mit Endezeichen "\0"), der eine ASCII-codierte, positive, ganze Dezimalzahl ($< 2^{32} = 4294967296$) beinhaltet. Schreiben Sie ein Assembler-Unterprogramm "AtouI", das

- beim Aufruf im Register R0 die Adresse des ersten Zeichens des Strings enthält und
- nach Beendigung des Unterprogramms steht im Register R0 die konvertierte 32-Bit-Zahl im Datenformat „unsigned integer“.

Beispiel:

Z. B. stehen ab Adresse 0x4000 0100 die Werte 0x36, 0x35, 0x35, 0x33, 0x35, 0x00, die dem String "65535", "\0" entsprechen und somit die Zahl 65535 darstellt.
- Beim Aufruf steht die Adresse in R0 = 0x4000 0100
- Nach Rückkehr aus dem Unterprogramm steht in R0 die Zahl 65535_D = 0x0000FFFF

Testen Sie Ihr Unterprogramm mit einem entsprechenden Hauptprogramm.

4.2 Funktion Berechnung von $Y = \left(\frac{2}{5}X\right)^2$

Schreiben Sie ein Unterprogramm "Berechnung" in Assembler, das von der Vorzeichen-behafteten, ganzen Zahl X (mit $X < 2^{16}$) den Funktionswert Y berechnet.

- Beim Aufruf des Unterprogramms steht im Register R0 die Zahl X.
- Nach Beendigung des Unterprogramms steht im Register R0 die Zahl Y.

Beispiel:

die Zahl 100_D = 0x0000 0064 → liefert 1600_D = 0x0000 0640

Testen Sie Ihr Unterprogramm mit einem entsprechenden Hauptprogramm.

4.3 Funktion uItoA (= unsigned Integer to ASCII)

Schreiben Sie ein Unterprogramm „uItoA“, das eine Vorzeichen-lose, ganze 16-Bit-Zahl vom Datentyp "unsigned integer" in einen ASCII-String (mit Endezeichen "\0") konvertiert. Dabei sollen führende Nullen unterdrückt werden.

Beim Aufruf steht im Register R0 die zu konvertierende Zahl und im Register R1 die Adresse für das erste Zeichen des zu erzeugenden Strings.

Beispiel:

die Zahl 65535_D = 0x0000FFFF
→ liefert den String 0x36, 0x35, 0x35, 0x33, 0x35, 0x00 und das entspricht "65535", "\0"

4.4 Hauptprogramm

Schreiben Sie ein Hauptprogramm, das:

- die Startadresse eines Strings STR_1 im **Datenspeicher** in das Register R0 lädt,
- dann die Funktion "AtouI" aufruft, die anschließend die konvertierte Zahl X zurückgibt,
- danach mit der konvertierten Zahl X im Register R0 die Funktion "Berechnung" aufruft, die den Funktionswert Y zurückliefert und
- dann mit dem Funktionswert im Register R0 und der Startadresse des zu erstellenden Strings STR_2 im Register R1 die Funktion "uItoA" aufruft, die den berechneten Funktionswert Y in einen ASCI-String mit "\0" als Endezeichen konvertiert.

4.4 主程序

写一个主程序，它：

* 将字符串STR_1在数据存储区的起始地址加载到寄存器R0中；
* 然后调用函数“AtouI”，该函数随后返回转换后的数字X；
* 之后，使用寄存器R0中转换后的数字X调用函数“Berechnung”，该函数返回函数值Y；
* 然后，使用寄存器R0中的函数值和将要创建的字符串STR_2的起始地址（在寄存器R1中）调
“uItoA”，该函数将计算出的函数值Y转换为以“\0”作为结束符的ASCII字符串。