Labor „Mikrocontroller“
SoSe 2024

Jan Bredereke
Hochschule Bremen

A: 20.6. / B: 27.6.2024

Laboraufgabe 6

Programmierung einer 32-Bit-Addition in Assembler

In dieser Aufgabe schreiben Sie einige kleine Programme in Assembler, die auf verschiedene Weisen eine 32-Bit-Addition ausführen. Ziel der Aufgabe ist, dass Sie lernen, ein sehr einfaches Assember-Programm zu schreiben, mithilfe der Entwicklungsumgebung zu übersetzen und ablaufen zu lassen und Fehler darin zu suchen und zu beheben. Über die verschiedenen Varianten des Assembler-Programms lernen Sie weiterhin die Anwendung mehrerer verschiedener Adressierungsarten, insbesondere die indirekte Adressierung und dabei auch den Zugriff auf den externen Speicher mit mehreren Data-Pointern.

Einen Laborbericht müssen Sie nicht schreiben. Als Ergebnis müssen Sie am Ende jeder der drei Teilaufgaben Ihr laufendes Programm sowie den dazugehörigen Quellcode Herrn Bredereke vorführen.

Vorübung: Ausführen einer fertigen Vorlage

In Aulis finden Sie das Assembler-Programm `add_direkt.a51`. Das Assembler-Programm enthält ein Unterprogramm `add_direkt`, das zwei 32-Bit-Zahlen addiert, indem es direkte Adressierung verwendet. Es erwartet den ersten Summanden im internen Speicher an den Adressen 40h bis 43h. Der zweite Summand muss entsprechend an den Adressen 50h bis 53h des internen Speichers stehen. Das Ergebnis wird vom Programm an die Speicherstellen des ersten Summanden geschrieben. Beachten Sie, dass das niederwertigste Byte (LSB) an der höchsten Adresse (z.B. 43h) und das höchstwertige Byte (MSB) an der niedrigsten Adresse (z.B. 40h) erwartet wird. Außerdem enthält das Assembler-Programm ein Hauptprogramm, das die Summanden bereitstellt und dann das Unterprogramm `add_direkt` aufruft. Zahl als LSB

- Legen Sie in der Entwicklungsumgebung ein Projekt an, fügen Sie die Assembler-Datei `add_direkt.a51` dazu, übersetzen Sie sie und führen Sie sie im Simulator oder ggf. nach Wunsch auf dem echten Mikrocontroller C515C aus.
- Überprüfen Sie das Ergebnis der Addition.
- Ändern Sie die Werte der Summanden im Hauptprogramm und führen Sie die Addition erneut aus.
- Ändern Sie die Werte der Summanden direkt im Simulator und starten Sie die Addition beim Aufruf des Unterprogramms.
- Starten Sie das Hauptprogramm wieder wie vorher am Anfang und arbeiten es im Einzelschrittbetrieb solange ab, bis der Aufruf des Unterprogramms erfolgt. Beobachten und interpretieren Sie die Veränderungen des Stackpointers und die Veränderungen des Stacks.
- Setzen Sie einen Unterbrechungspunkt (Breakpoint) auf die Programmadresse, an der der Unterprogrammaufruf steht. Führen Sie ab da das Programm in Einzelschrittmodus aus.

1. Indirekte Adressierung internen Speichers über Register

- Kopieren Sie das obige Programm in eine Datei `add_indirekt.a51`, benennen Sie das Unterprogramm ebenfalls entsprechend um und legen Sie zur neuen Datei ein weiteres Projekt an.
- Ändern Sie den Code so, dass dem Unterprogramm nun als Parameter die Adresse des niederwertigsten Bytes (LSB) des ersten Summanden bzw. des Ergebnisses in Register R0 übergeben wird und die Adresse des LSB des zweiten Summanden in Register R1 übergeben wird.
- Ändern Sie den Code weiterhin so, dass die vier Teiladditionen im Unterprogramm durch einen einzigen Additionsbefehl in einer Schleife realisiert werden.
- Das Unterprogramm soll auch mit einem beliebigen anderen Hauptprogramm funktionieren können, sofern dieses als Parameter genau die beiden Register R0 und R1 übergibt.
- Testen Sie Ihr neues Unterprogramm, unter anderem mit den Summanden aus der Vorübung, und beheben Sie ggf. Fehler. Kopieren Sie dabei den Teil des Hauptprogramms, der die Parameter setzt und `add_indirekt` aufruft, so dass er gedoppelt ist, und tragen Sie in die Kopie eigene Summanden und *andere Adressen* dafür ein (d.h. nicht sum1 und sum2).

2. Adressierung externen Speichers über 8-Bit-Register

- Kopieren Sie das Programm aus der vorigen Teilaufgabe in eine Datei `add_extern.a51`, benennen Sie das Unterprogramm ebenfalls entsprechend um und legen Sie zur neuen Datei ein weiteres Projekt an.
- Ändern Sie den Code so, dass die Summanden nun nicht im internen Speicher, sondern im externen Speicher liegen, und zwar im Adressbereich zwischen 0h und 0ffh (8-Bit-Adressen).

  Beachten Sie, dass unsere Mikrocontroller über volle 64 kByte Speicher verfügen, und dass daher die 8-Bit-Adressierung nur einen Teilbereich des vorhandenen Speichers erreicht. Die oberen 8 Bit der resultierenden Adresse werden durch den Ausgabeport P2 bestimmt. Dieser enthält nach einem Systemstart zunächst den Wert 0ffh und muss daher vor einem Zugriff auf den externen Speicher passend initialisiert werden.
- Das Unterprogramm soll wiederum auch mit einem beliebigen anderen Hauptprogramm funktionieren können, sofern dieses als Parameter genau die beiden Register R0 und R1 übergibt.
- Testen Sie Ihr neues Unterprogramm, unter anderem mit den Summanden aus der Vorübung, und beheben Sie ggf. Fehler. Kopieren Sie dabei den Teil des Hauptprogramms, der die Parameter setzt und `add_extern` aufruft, so dass er gedoppelt ist, und tragen Sie in die Kopie eigene Summanden und *andere Adressen* dafür ein (d.h. nicht sum1 und sum2).

3. Adressierung externen Speichers über 16-Bit-Register

- Kopieren Sie das Programm aus der vorigen Teilaufgabe in eine Datei `add_dptr.a51`, benennen Sie das Unterprogramm ebenfalls entsprechend um und legen Sie zur neuen Datei ein weiteres Projekt an.

- Ändern Sie den Code so, dass die Summanden nun an beliebigen Stellen im externen Speicher liegen können, d.h. im Adressbereich zwischen 0h und 0ffffh (16-Bit-Adressen).

  Nutzen Sie aus, dass der Mikrocontroller C515C nicht nur über einen, sondern über acht Datapointer (DPTR) verfügt, von denen jeweils einer als aktiver Datapointer ausgewählt werden kann. Dies kann mithilfe des Special-Function-Registers DPSEL geschehen, vergleiche den Foliensatz und das Notizen-Dokument „Die Strukturerweiterungen des C515C-Mikrocontrollers“, Abschnitt „Data-Pointer“.

  Dem Unterprogramm soll als Parameter in DPTR0 die Adresse des LSB des ersten Summanden und des Ergebnisses übergeben werden, und in DPTR1 die Adresse des LSB des zweiten Summanden.

  Beachten Sie, dass es zwar einen Maschinenbefehl gibt, um den Datapointer zu inkrementieren, aber aber keinen Maschinenbefehl, um ihn zu dekrementieren, und dass weiterhin die Anordung der Bytes in den 32-Bit-Zahlen ein Dekrementieren verlangt. Realisieren Sie das Dekrementieren daher mit Hilfe eines eigenen Unterprogramms, das den 16-bittigen DPTR über die beiden 8-bittigen Special-Function-Register DPL und DPH geeignet manipuliert.

- Testen Sie Ihr neues Unterprogramm, unter anderem mit den Summanden aus der Vorübung, und beheben Sie ggf. Fehler.

  Testen Sie insbesondere mit den Adressen DPTR0=2002h (d.h. Daten von 1fffh bis 2002h) und DPTR1=2012h sowie mit den Adressen DPTR0=3001h und DPTR1=3011h, und zwar in beiden Fällen mit den 32-Bit-Zahlen aus der obigen Vorlage.