Lehrstuhl Systemsicherheit
Prof. Dr. Thorsten Holz
Dipl.-Inf. Ralf Hund
Dipl.-Inform. Carsten Willems

Vorlesung Programmanalyse
Herbst-/Wintersemester 2011/2012
Ruhr-Universität Bochum

1. Übung

27. Oktober 2011
Abgabe der Hausaufgaben: per Moodle bis zum 10. November 2011
Bei Fragen und Problemen können Sie sich per E-Mail/Moodle an die Tutoren wenden.

Aufgaben

In den nächsten Aufgaben sollen Sie sich mit den Grundlagen der Programmierung in Intel Assembler vertraut machen. Dafür wird der Microsoft Macro Assembler (MASM), genauer gesagt das MASM32-Paket, verwendet. Das entsprechende Paket kann auf der Vorlesungsseite in Moodle heruntergeladen werden. Im Zip-Archiv (*masm32v10-sre.zip*) befinden sich auch zwei simple Beispiele, die als Template verwendet werden können, und eine Batch-Datei zum Aufruf des Assemblers. Falls Sie MASM32 in ein anderes Verzeichnis als das Standardverzeichnis installieren, dann muss der entsprechende Pfad in der Batch-Datei angepasst werden. Der erste Parameter ist die Eingabedatei (.asm), der zweite Parameter ist das sog. Subsystem. Für Konsolenanwendungen muss dies *CONSOLE* sein, ansonsten *WINDOWS*.

Aufgabe 1: Ein- und Ausgabe in Assembler (1 Punkt)

Schreiben Sie ein Konsolenprogramm in Assembler, das drei Zahlen als Benutzereingabe einliest, die ersten beiden Zahlen addiert, die dritte Zahl von der Summe subtrahiert und das Ergebnis der Berechnungen ausgibt.

Hinweise: Sie können für die Interaktion mit der Konsole Funktionen aus der Standard-C Library verwenden. Im Internet existieren dafür diverse Dokumentationen. Es bietet sich an, `printf` zur Textausgabe und `scanf` zur Eingabe zu verwenden. Weiterhin könnte im weiteren Verlauf die Funktion `strlen` hilfreich sein. In MASM muss jeder Standard-C Funktion ein `crt_` vorangestellt werden (also z.B. `crt_printf`, `crt_scanf`, etc.).

Im Übungsordner befindet sich eine Datei names *io_example.asm* mit einem Beispiel zur Ein- und Ausgabe mittels `crt_printf` und `crt_scanf`.

Aufgabe 2: Sprünge und bedingte Verzweigungen in Assembler (1 Punkt)

Schreiben Sie ein Konsolenprogramm in Assembler, welches als Taschenrechner für Addition, Subtraktion und Multiplikation verwendet werden kann. Dazu wird nach dem Start des Programmes der Benutzer befragt, welche Rechenoperation durchgeführt werden soll (Eingabe: 0 = Addition, 1 = Subtraktion, 2 = Multiplikation). Bei fehlerhafter Eingabe (> 2) soll eine entsprechende Fehlermeldung ausgegeben und das Programm beendet werden. Ist die Operationseingabe erfolgreich gewesen, sollen zwei Zahlen eingegeben werden können, welche dann mit der gewählten Rechenoperation verrechnet werden.

Aufgabe 3: Schleifen und Register in Assembler (2 Punkte)

Schreiben Sie ein Konsolenprogramm in Assembler, welches die Quersummen über die ASCII-Werte der eingegebenen Zeichenketten berechnet. Der Benutzer soll eine Zeichenkette eingeben, von welcher die Quersumme gebildet wird. Beispielsweise ist die Quersumme des ASCII-Strings „0123456789“ $= 48 + 49 + 50 + \cdots + 57 = 525$, die von „abc“ $= 97 + 98 + 99 = 294$. Verwenden Sie für die Quersummenberechnung keine lokalen oder globalen Variablen als Zwischenspeicher, sondern arbeiten Sie ausschließlich auf Registern.

Aufgabe 4: Analyse von Programmen I (2 Punkte)

1. Analysieren Sie das Programm *wisdom.exe*. Laden Sie dazu die Datei in einen Debugger Ihrer Wahl (z.B. OllyDbg). Achten Sie bei Ihrer Analyse besonders auf die API Calls und den Kontrollfluss nachdem die Benutzereingabe eingelesen wurde. Was verlangt das Programm von Ihnen, damit die Erfolgsnachricht ausgegeben wird? Geben Sie das Lösungswort in Moodle ein. Sie haben maximal 3 Versuche, das korrekte Lösungswort einzugeben. (1 Punkt)

2. Übersetzen Sie das Programm in die Programmiersprache C und geben Sie den Quelltext an. (1 Punkt)

Aufgabe 5: Analyse von Programmen II (3 Punkte)

1. Analysieren Sie das GUI Programm *ControlSystem.exe*. Gehen Sie dabei wie in der vorherigen Aufgabe vor, beachten Sie jedoch, dass in diesem Fall WinAPI Calls verwendet werden, um Text einzulesen und auszugeben. Finden Sie einen gültigen Schlüssel für den Benutzer *Bob*. Ist es nötig den Algorithmus zu verstehen, um einen gültigen Schlüssel zu finden? Begründen Sie! (1 Punkt)

2. Geben Sie eine Liste von 10 gültigen Benutzer/Schlüsselkombinationen an. Implementieren Sie dazu ein Programm, das die nötigen Rechnungen durchführt. Bitte geben Sie Ihren Programmquelltext mit an, wobei Sie eine beliebige Hochsprache benutzen können (also z.B. C++, Python, PHP, aber kein Assembler). (2 Punkte)

Aufgabe 6: Analyse von Programmen III (5 Punkte)

1. Analysieren Sie das Programm *chal.exe* mit einem Disassembler oder Debugger Ihrer Wahl und patchen Sie die ausführbare Datei derart, dass das Programm immer eine Erfolgsmeldung ausgibt - unabhängig davon, ob die eingegebene Lösung richtig ist oder nicht. Bitte beschreiben Sie kurz in 2 Sätzen wie Sie vorgegangen sind (welche Speicherstellen gepatcht werden müssen). (1 Punkt)

2. Im nächsten Schritt soll das Programm genauer analysiert werden. Geben Sie die passende Lösung für die Challenge „ReverseEngineer“ an! (1 Punkt)

3. Sollten Sie auch diese Aufgabe meistern, dann versuchen Sie einen eigenen Generator zu schreiben, der für eine beliebige Challenge als Eingabe die entsprechende Lösung generiert. Bitte benutzen Sie eine Hochsprache Ihrer Wahl (also z.B. C++, Python, PHP, aber kein Assembler) und geben Sie ihren Quellcode ab. (2 Punkte)

4. Das Programm enthält zusätzlich eine Schwachstelle, sodass es bei bestimmten Challenges jede Lösung akzeptiert. Wo befindet sich diese Schwachstelle (Hint: Auch viele C Programme leiden unter solchen Schwachstellen)? Welche Eingaben werden immer akzeptiert und wie kann man die Schwachstelle reparieren? (1 Punkt)

Abgabe der Übungen

Die Hausaufgaben sind bis zur übernächsten Woche in Moodle einzureichen. Sie erreichen die Moodle-Seite per `http://moodle.rub.de`: Loggen Sie sich in das System ein und wählen Sie unter *Meine Kurse* die Veranstaltung *Programmanalyse WS 11/12* aus.