

Sichere Programmierung

Projekt 2

Julian Sobott
(76511)

David Sugar
(76050)

Inhaltsverzeichnis

1	Zu Aufgabe 1	3
1.1	a)	3
1.2	b)	3
1.3	c)	3
1.4	d)	4
2	Zu Aufgabe 1	7
2.1	a)	7
2.2	b)	7
2.3	c)	8
2.3.1	Reihenfolge der Funktionsaufrufe	8
3	Zu Aufgabe 3	9
3.1	a) Analysieren Sie den in der Datei enthaltenen Source Code	9
3.1.1	Implementierung	9
3.1.2	Aufruf	10
3.2	b) Kompilieren Sie den C Code und führen Sie das Programm aus	10
3.2.1	Kompilieren	10
3.2.2	Ausführen	10
3.3	c) Führen Sie das Programm im GDB aus	11
3.3.1	Wie viele Stack Frames werden erzeugt?	11
3.3.2	Wie ist der Inhalt dieser Stack Frames?	11
3.3.3	Wie wird die Parameterübergabe in Assembler umgesetzt?	11

1 Zu Aufgabe 1

1.1 a)

Zu Beginn der `main()` Funktion wird eine `unsigned int` Variable, `i`, deklariert, jedoch nicht initialisiert, d.h. bis auf wenige Ausnahmen $i \in \{0..2^{32} - 1\}$.

Danach wird die Variable im Kopf der darauf folgenden For-Schleife mit 0 initialisiert. Die Schleife inkrementiert die Variable `i` am Ende jedes Schleifendurchlaufs und tritt erneut in die Schleife ein, solange `i` kleiner 20 ist. Innerhalb der Schleife wird der Wert von `i`, zum jeweiligen Zeitpunkt, formatiert mithilfe von `printf()` in der Standardausgabe ausgegeben. Dabei werden immer 2 Stellen ausgegeben, dies wird über `"%2d"` realisiert.

Potentielles Problem: Es sollte `"%2u"` verwendet werden, da `d` für die Formatierung von signed Integeren verwendet wird. In diesem Fall spielt die Formatierung aber keine Rolle.

1.2 b)

Bild 1 zeigt die Ausgabe des Programms.

1.3 c)

```
1      <+8>:   mov     DWORD PTR [rbp-0x4],0x0
2      <+15>:  jmp     0x40113b <main+41>
3      <+17>:  mov     eax,DWORD PTR [rbp-0x4]
4      <+20>:  mov     esi,eax
5      <+22>:  mov     edi,0x402004
6      <+27>:  mov     eax,0x0
7      <+32>:  call    0x401030 <printf@plt>
8      <+37>:  add     DWORD PTR [rbp-0x4],0x1
9      <+41>:  cmp     DWORD PTR [rbp-0x4],0x13
10     <+45>:  jbe     0x401123 <main+17>
```

Für die Variable `i` wird Speicher auf dem Stack alloziert, die Anfangsadresse ist dabei `rbp-0x4`.

In Zeile `<+8>` wird `i` mit `0x0` initialisiert. Danach springt das Programm unbedingt in Zeile `<+41>`. Hier befindet sich nun die Überprüfung, ob die Schleife verlassen wird, d.h. $i \geq 0x14$, oder ein weiterer Schleifendurchlauf gestartet wird. Dazu wird in Zeile `<+41>` `i` mit `0x13` verglichen. Ist der Wert kleiner oder gleich `0x13` wird in Zeile `<+17>` gesprungen und damit ein weiterer Schleifendurchlauf gestartet. Andernfalls wird die nächste Instruktion ausgeführt und damit die Schleife verlassen.

In Zeile `<+17>` und `<+20>` wird der Wert von `i`, vom Speicher in das `esi` Register geladen. In der darauf folgenden Zeile wird die Adresse des Formatierungsstrings ("`i: %2d n`") (`0x402004`) in `edi` geladen.