Name der Vorlesung
Weiterführende Aufgaben zu ARM Assembly

1 Saying Hello

In den nachfolgenden Aufgaben geht es darum, Ihre Kenntnisse über ARM Assembly und Linux system calls anzuwenden, die Sie durch die Vorlesung gewonnen haben.

Die Aufgaben bauen auf einander auf, d.h. Sie sollten die Aufgaben in der Ihnen vorgegebenen Reihenfolge bearbeiten.

Eine Auflistung aller Linux system calls findet sich hier.

1.1 Aufgabe 1: Hello ARM

Das "Hello, World!" Programm ist in der Regel das erste Programm, das ein Programmierer schreibt, wenn er sich an eine neue Programmiersprache wagt. Wir wollen diese Tradition fortführen.

Erstellen Sie ein ARM Assembler-Programm, dass "Hello, ARM!" auf der Kommandozeile ausgibt und sich danach ordnungsgemäß, d.h. mit Rückgabewert 0, beendet.

Beispielausgabe

```
Hello, ARM!
```

Einschränkung: Nutzen Sie nicht den `ldr <Rd>, =imm` Pseudobefehl.

1.2 Aufgabe 2: strlen()

Die Länge eines Strings dynamisch zur Laufzeit bestimmen zu können ist nützlich, da so Strings ausgegeben werden können ohne deren Länge im Vorfeld wissen zu müssen.

Erweitern Sie das Programm aus Aufgabe 1) und schreiben Sie eine Funktion namens **strlen**, die einen `NULL terminierten` String als Argument übergeben bekommt und dessen Länge, ausschließlich **Nullterminator**, zurückliefert.

```
unsigned strlen(char* str)
```

Testen sie die Funtion mittels GDB. Nutzen Sie strlen um die länge von "Hello, ARM!" dynamisch zu berechnen.

Beispielausgabe

```
Hello, ARM!
```

1.3 Aufgabe 3: puts()

Kapseln Sie den Code zur Ausgabe des Strings in einer neuen Funktion namens **puts**. Die Funktion soll die Adresse eines beliebigen NULL terminierten Strings als argument übergeben bekommen und diesen dann über **stdout** ausgeben.

```
void puts(char* str)
```

Tip: Für eine Übersicht über alle ASCII Symbole siehen `man ascii`.

Beispielausgabe

```
Hello, ARM!
```

1.4 Aufgabe 4: strncpy()

Strings kopieren zu können ist ein wichtiges Feature jeder Programmiersprache und auch für die folgenden Übungen wäre eine Kopierfunktion durchaus nützlich. Implementieren Sie eine Funktion namens **strncpy**, die als erstes Argument die Adresse eines Zielstrings erhält, als zweites Argument die Adresse des zu kopierenden Strings und als drittes Argument die Anzahl an Bytes die es zu kopieren gilt. Die Funktion soll die Adresse des Zielstrings zurückliefern.

```
char* strncpy(char *dest, char *src, unsigned n)
```

Vergewissern Sie sich, dass der dest String immer NULL terminiert ist, d.h. dest[n-1] = 0.

Testen Sie die Funktion mittels GDB. Kopieren Sie einen String und geben Sie diesen auf der Kommandozeile aus. Experimentieren Sie mit verschiedenen Eingabelängen für strncpy.

Beispielausgabe

```
fÜr n = 6 und src = "Hello, ARM!"

Hello
```

1.5 Aufgabe 5: puts() revisited

Die in Aufgabe 1.3) implementierte Funktion puts() unterscheidet sich in einem entscheidenden Aspekt von der Version, die man aus der Programmiersprache C kennt. Dort gibt puts nicht nur den String aus, sondern hängt auch noch einen line feed (siehe man ascii) an das Ende des Strings, wodurch es auf der Kommandozeile zu einem Zeilenumbruch kommt.

Erweitern Sie Ihre Funktion, sodass diese das gleiche Verhalten aufweist.

Einschränkung: Der write Syscall soll nur ein einziges mal aufgerufen werden.

TIP: Versuchen Sie den auszugebenden String auf dem Stack vorzubereiten.

1.6 Aufgabe 6: Hello Me

Zum Schluss soll das Programm noch so erweitert werden, dass ein beliebiger String der Länge <= n, für eine beliebige statische Zahl n, eingelesen und auf der Kommandozeile wieder ausgegeben werden kann. Schreiben Sie eine Funktion getsn, die die Adresse eines Buffers, sowie dessen größe überbegeben bekommt und eine Zeile von der Standardeingabe einließt. Die Funktion stopt sobald das Ende der File erreicht ist (eof) oder n-1 Zeichen eingelesen wurden. Das newline Symbol soll nicht mit in den Buffer geschrieben werden. Der Buffer soll Null-terminiert sein. Die Funktion soll die Anzahl gelesener Bytes zurück geben.

```
int getsn(char* buf, int n)
```

Beispielausgabe

```
\$ What is your name? >>> Bob
Hello, Bob!
```