LEHRSTUHL FÜR RECHNERTECHNIK UND RECHNERORGANISATION

Rechnerarchitekturpraktikum

Howto: Kompilieren mit GCC und NASM

Um Ihnen einen leichteren Einstieg in Ihr Assembler-Projekt zu ermöglichen, zeigt Ihnen dieses Howto, wie man eine Assembler-Routine aus einem C-Programm heraus aufruft.
Bitte legen Sie sich bereits bei der Erstellung des Pflichtenhefts auf die Zielplattform, sowohl hardware- (x86 oder x64) als auch softwareseitig (Windows, Linux oder Mac OS), fest und erwähnen Sie diese dort, da sich die Vorgehensweise jeweils unterscheidet.
Diese Anleitung wurde getestet auf

- Ubuntu 16.10 x64,
- Mac OS X 10.11 und
- Windows 10 x64 mit msys2.

Sollten Sie einen Fehler finden oder trotz dieser Anleitung Probleme mit dem Kompilieren haben, wenden Sie sich bitte (rechtzeitig) an Ihren Tutor.

Inhaltsverzeichnis

1 Vorgehensweise auf x86

- Erstellen Sie eine Datei, die Ihre Assembler-Routine beinhaltet.

```
; Es folgt 32-Bit-Assembler.
BITS 32

; Das Label (= Funktionsname) soll als Symbol exportiert
    werden.
GLOBAL doIt

doIt:
    mov eax, 42
ret
```

Listing 1: routine.asm

- Schreiben Sie ein C-Programm, das Ihre Assembler-Routine aufruft.

```
#include <stdio.h>

// Dem Compiler mitteilen, dass sich die Funktion "doIt" in
    einer anderen Datei befindet.
extern int doIt() asm("doIt");

int main() {
    printf("%d\n", doIt());
}
```

Listing 2: rahmenprogramm.c

- Übersetzen Sie nun zunächst Ihre Assembler-Routine in eine Objektdatei.

```
linux:~> nasm -f elf32 -o routine.o routine.asm

windows:~> nasm -f elf32 -o routine.o routine.asm
```

- Nun übersetzen Sie Ihr C-Programm und geben die vom Assembler generierte Objektdatei als weitere Eingabedatei an.

```
linux:~> gcc -m32 -o programm rahmenprogramm.c routine.o

windows:~> gcc -m32 -o programm rahmenprogramm.c routine.o
```

- Testen Sie das erstellte Programm.

```
pc:~> ./programm
42
pc:~>
```

- Bevor Sie mit der Implementierung Ihrer eigentlichen Lösung beginnen, informieren Sie sich insbesondere
  - über den Aufbau eines Stackframes in C und
  - über die auf Ihrer Architektur verwendete Calling Convention (cdecl).

2 Vorgehensweise auf x64

- Erstellen Sie eine Datei, die Ihre Assembler-Routine beinhaltet.

```
; Es folgt 64-Bit-Assembler.
BITS 64

; Das Label (= Funktionsname) soll als Symbol exportiert
    werden.
GLOBAL doIt

doIt:
    mov rax, 42
ret
```

Listing 3: routine.asm

- Schreiben Sie ein C-Programm, das Ihre Assembler-Routine aufruft.

```
#include <stdio.h>

// Dem Compiler mitteilen, dass sich die Funktion "doIt" in
    einer anderen Datei befindet.
extern int doIt() asm("doIt");

int main() {
    printf("%d\n", doIt());
}
```

Listing 4: rahmenprogramm.c

- Übersetzen Sie nun zunächst Ihre Assembler-Routine in eine Objektdatei.

```
linux:~> nasm -f elf64 -o routine.o routine.asm

windows:~> nasm -f elf64 -o routine.o routine.asm

mac:~> nasm -f macho64 -o routine.o routine.asm
```

- Nun übersetzen Sie Ihr C-Programm und geben die vom Assembler generierte Objektdatei als weitere Eingabedatei an.

```
linux:~> gcc -m64 -o programm rahmenprogramm.c routine.o

windows:~> gcc -m64 -o programm rahmenprogramm.c routine.o

mac:~> clang -m64 -o programm rahmenprogramm.c routine.o
```

- Testen Sie das erstellte Programm.

```
pc:~> ./programm
42
pc:~>
```

- Bevor Sie mit der Implementierung Ihrer eigentlichen Lösung beginnen, informieren Sie sich insbesondere
  - über den Aufbau eines Stackframes in C und
  - über die auf Ihrer Architektur verwendete Calling Convention (System V AMD64 ABI).