Technische Hochschule Ingolstadt
Fakultät Informatik
Prof. Dr. Nikoletta Sofra
Wintersemester 2025-26

Praktikum Rechnerarchitektur - Informatik BSc - Blatt 1

Erste Schritte, Hello World!

Wir verwenden dieselbe virtuelle Maschine wie in GdP1
– Lars lässt grüßen.

Wir verwenden den nasm **Assembler.**
Hier ist die offizielle Dokumentation in HMTL:
https://www.nasm.us/xdoc/2.16.03/html/nasmdoc0.html

Wir verwenden gcc zum **Linken.**

Zunächst testen wir, dass alles funktioniert - wir rufen Im Terminal die Version unserer Tools auf:

```
$ nasm -v
```

```
$ gcc -v
```

Mit

```
$ nasm -h
```

werden alle Optionen des nasm Befehls erläutert (= das, was wir in Command Line schreiben und vom Assembler anfordern dürfen).

Kopieren Sie die Quelldatei `HelloWorld.asm` UND die Datei `Lab1.asm` (die werden wir später brauchen) in Ihr Arbeitsverzeichnis (am besten ein Verzeichnis für RAP/Lab1 erstellen).

Nun assemblieren Sie die Quelldatei `HelloWorld.asm` mit dem **nasm** :

```
$ nasm -f elf32 HelloWorld.asm -l HelloWorld.lst
```

Was machen die Optionen -f unf -l? Verwenden Sie

```
$ nasm -h
```

um das herauszufinden!

Überprüfen Sie die Inhalte Ihres Arbeitsverzeichnisses! Welche neuen Dateien sehen Sie? Was sind diese?

Schauen Sie in der Listing-Datei rein:

`$ vi HelloWorld.lst` oder `$ cat HelloWorld.lst`

Notiz: Die Listing-Datei (HelloWorld.lst) benötigen wir nicht immer, wenn wir assemblieren. Das wird hier erwähnt und genutzt, um das Ergebnis der Assemblierung darzustellen.

Jetzt müssen wir linken!

```
$ gcc HelloWorld.o -o HelloWorld.out
```

Um das Programm aufzuführen, schreiben Sie:

```
$ ./HelloWorld.out
```

```
lars@ws23-AsmCpp2204-32bit:~/RAP/RAP/Lab1
$ nasm -f elf32 HelloWorld.asm -l HelloWorld.lst
lars@ws23-AsmCpp2204-32bit:~/RAP/RAP/Lab1
$ gcc HelloWorld.o -o HelloWorld.out
/usr/bin/ld: warning: HelloWorld.o: missing .note.GNU-stack section implies executable stack
/usr/bin/ld: NOTE: This behaviour is deprecated and will be removed in a future version of the linker
/usr/bin/ld: HelloWorld.o: warning: relocation in read-only section `.text'
/usr/bin/ld: warning: creating DT_TEXTREL in a PIE
lars@ws23-AsmCpp2204-32bit:~/RAP/RAP/Lab1
$ ./HelloWorld.out
Hello World
```

Die erste Aufgabe

Nun schauen wir in die Datei `Lab1.asm` rein. Was macht dieses Programm?

Assemblieren, linken und führen Sie dieses Programm auf. Was passiert?

Jetzt schreiben Sie das Programm um:

- **Die Werte 21 und 2 sollen aus zwei Speichervariablen mit den symbolischen Adressen a und b gelesen werden.**
- **Das Ergebnis soll in HEX Format erscheinen**

Während Sie das Programm umbauen, ist es empfohlen, einen Debugger zu nutzen, zum Beispiel GDB. Dafür müssen wir dem Assembler instruieren, damit die vom Debugger benötigte Information der Output-Datei angehängt wird. Dafür verwenden wir die Option `-g` . Die Format wird mit `-F` definiert:

```
nasm -g -f elf32 -F dwarf MyProgramme.asm
```

Um die neuen Variablen zu definieren, müssen Sie in dem Data Segment zwei neu Variablen hinzufügen:

```
<var> : <type> <value>
```

Verwenden Sie als `<type> dd` (das definiert ein Double Word = 4 Bytes).

`var` ist eigentlich ein Zeiger, bzw die Adresse, wo die Variabel gespeichert ist. Um die Variabel in einem Register <Reg> zu kopieren, schreiben wir:

`mov <Reg> [var]` (Zum Beispiel `mov eax, [var]`)

Die Variabel, die dem `printf` Befehl für die HEX Format übergeben werden muss, ist schon als string deklariert. Wo muss die hin?

Jetzt erweitern Sie dieses Programm, um das Skalarprodukt zwei Ortsvektoren x = <x1. x2, x3> und y =<y1, y2, y3> zu berechnen. Verwenden Sie x = <17, 11, 4> und y = <4, -9, 8>

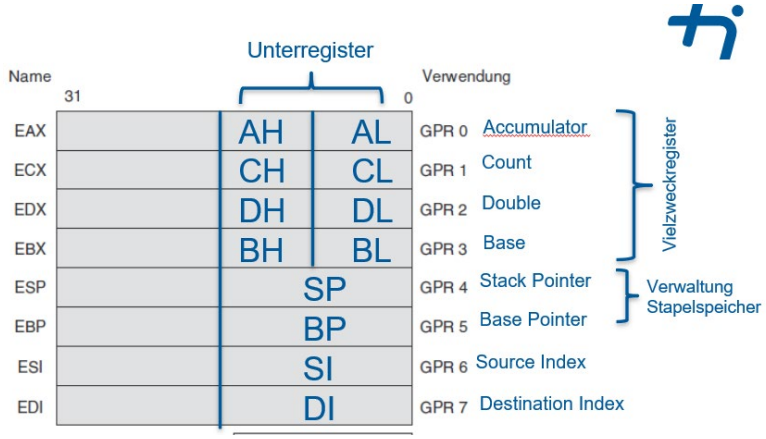
Definieren Sie die Vektoren als dreidimensionale Doppelwort Speichervariablen.

Verwenden Sie den Befehl

```
imul r1_32/r2_32 : r1_32 x r2_32 → r1_32
```

und geben Sie das Ergebnis mit `printf` aus.

Hier nochmal der Registersatz der zur Verfügung steht:



Den Befehlssatz finden Sie auf die Vorlesungsunterlagen (oder online 😊)

Wenn das noch nicht der Fall ist, schreiben Sie die Berechnung des Skalarprodukts als Iteration (Loop).

Hier wird als Beispiel ein Loop für 10 Iterationen gezeigt:

```
      mov ecx, 10
next:     …
      …
      …
      loop next
```

Oder eine alternative Variante:

```
      mov ecx, 0
next:     …
      …
      …
      inc ecx
      cmp ecx, 10
      jne next
```

Verwendung von dem Debugger gdb mittels TUI (Text User Interface)

Es muss wie folgt assembliert werden:

```
nasm -g -f elf32 -F dwarf MyProgramme.asm
```

und dann (wie immer) gelinkt werden:

```
gcc MyProgramme.o -o MyProgramme.out
```

Starten Sie das TUI:

```
gdbtui ./MyProgramme.out
```

Sie können alle Register separat in einem Fenster zeigen lassen mit:

```
layout regs
```

Definieren Sie ein Breakpoint.

Wenn `Label` der Name einer Befehlsaddresse ist, dann:

```
break Label
```

Starten Sie das Programm: `run`

Weitermachen: `ni` (next instruction) `oder c` (continue)

Hier ein paar Vorschläge zum Probieren:

Inhalte zeigen lassen mit p (steht für Print):

Inhalt von `eax` dargestellt als Character (ASCII): `p /c $eax`

Inhalt von `eax` dargestellt im Oktalsystem: `p /o $eax`

Andere Formate:

`/a` (Addresse), `/d` (signed), `/u` (unsigned),
`/f` (floating point), `/t` (binary)

Speicherinhalte untersuchen mit x (steht für eXamine)

Das ist sinnvoll, um das Stack zu untersuchen, zB:

- Zeige die obere 2 Werte als hexadezimale Zahlen: `x /2x $esp`
- Zeige das oberste Element (eine Adresse): `x /a $esp`

Für das erste Testat müssen Sie bis Ende KW45:

- **Als Team ausschließlich in den Praktikumszeiten Ihrer Gruppe:**
  - Ihr Programm mit der Berechnung des Skalarprodukts als Iteration auf einem Laborrechner zeigen und Fragen dazu beantworten können
  - Ihr Programm assemblieren, linken und vorführen
  - Den Debugger starten und in jeder Iteration die Inhalte eines gewählten Registers zeigen (Notizen sind erlaubt)
- **Das Programm mit der Berechnung des Skalarprodukts als Iteration ins Moodle hochladen** (oben in den Kommentaren schreiben Sie bitte mit wem Sie im Team zusammenarbeiten).
  Es ist wichtig, dass alle Studierende das Programm separat hochladen, damit ich die Testatvergabe in Moodle bestätigen kann!