Hochschule RheinMain
Fachbereich Design Informatik Medien
Marcus Thoss, M.Sc.

# Mikroprozessortechnik
# SS 2020
# LV 2522

# Übungsblatt 2

## Aufgabe 2.1 (Funktionsaufrufe in AVR-Assemler):

In diesem Aufgabenteil üben Sie, einfache Funktionsaufrufe in Assembler zu realisieren.

a) Kopieren Sie die Vorlage, wie in Aufgabenblatt 1, nun in eine Datei `mpt1b_regs.S` und fügen Sie ein Sprunglabel `addproc:` für eine neue Funktion am Ende hinzu.

b) Implementieren und kommentieren Sie diese Funktion, die zwei in den Registern `r24` und `r25` übergebene Parameter addieren und das Ergebnis in `r24` zurückliefern soll. Der Rücksprung aus der Funktion erfolgt mit dem Assemblerbefehl `ret`. Erweitern Sie Ihr Makefile und Skripte entsprechend.

c) Fügen Sie das Laden von Beispielparametern in `r24` und `r25` und einen Funktionsaufruf von `addproc` (mit `rcall`) im Hauptprogramm (vor `endloop`) ein und testen Sie Ihre Implementierung mit dem Simulator.

d) Versionieren Sie Ihre neuen Dateien (diese Anforderung wird in zukünftigen Aufgaben stillschweigend vorausgesetzt).

## Aufgabe 2.2 (ggT):

a) Designen und implementieren Sie mit dem Wissen der vorherigen Übungsteile eine rekursive Funktion `ggt(a,b)`, die den größten gemeinsamen Teiler von `a` und `b` ($\in \mathbb{N}$) nach folgendem Algorithmus von Euklid zurückliefert:

- falls $a = b$, dann ist $ggT(a, b) = a$.
- falls $a < b$, dann wende den Algorithmus $ggT$ an auf $(a, b - a)$.
- falls $b < a$, dann wende den Algorithmus $ggT$ an auf $(a - b, b)$.

b) Erstellen Sie aus der Assembler-Vorlage eine neue Datei `ggt.S` und implementieren und testen Sie die rekursive ggT-Funktion mit Übergabe der Paramter in `r24` und `r25` sowie Rückgabe in `r24`.

Beachten Sie, dass der Aufrufer der Funktion nicht annehmen darf, dass `r25` und r24 den Funktionsaufruf unverändert "überleben". Hinweis dazu: das Kopieren eines Wertes von einem Register in ein anderes erfolgt mit dem Befehl `MOV`.

## Aufgabe 2.3 (Funktionsaufrufe über den Stack):

In diesem Aufgabenteil betrachten Sie Funktionsaufrufe mit Parameterübergabe über den Stack.

a) Erstellen Sie eine neue Variante des Programms aus Aufgabe 1.2, `mpt1b_stack.S`. Die Übergabe von Parametern und Rückgabewert soll nun nicht mehr in Registern, sondern auf dem Stack erfolgen.

Dabei entstehen folgende neue Anforderungen:

- Vor dem Aufruf werden die beiden Parameter auf den Stack gelegt (`push`).
- Die Funktion muss die Parameter vom Stack holen. Skizzieren Sie, wie der Stack beim Aufruf aussieht, beobachten Sie ggf. im Simulator. Was liegt noch vor den Parametern auf dem Stack? Sichern Sie alle Informationen, die Sie vom Stack abräumen müssen, in Registern.
- Vor dem Rücksprung müssen nun der Rückgabewert und ggf. weitere für den Rücksprung gesicherte Informationen wieder auf den Stack gelegt werden; anschließend folgt ein `ret`.
- Laden Sie im Hauptprogramm nach der Rückkehr der Funktion das Ergebnis vor `endloop` noch in `r24`.

b) Modifizieren Sie die letzte Variante erneut zu `mpt1b_stackframe.S`. Nun soll in der Funktion das Abholen der Parameter vom Stack nicht mehr mit `pop` erfolgen!

Laden Sie stattdessen das Z-Register (r30/r31) mit dem aktuellen Stackpointer-Zeigerwert, um dann auf die im Stackspeicher liegenden Prameter direkt mit `ldd` (Adressierungsmodus "Data Indirect with Displacement") zuzugreifen. Zeichnen Sie sich ein Diagramm der Lage der Parameter und der Rücksprungadresse auf dem Stack, um Ihren Code zu planen.

Analog soll der Rückgabewert im Stackspeicher mittels `std` und Z-Register den letzten Eingabeparameter überschreiben (statt `push` zu verwenden).

Beachten Sie schließlich, dass der *Aufrufer* jetzt dafür verantwortlich ist, nach der Rückkehr der Funktion den Stack von den Parametern zu befreien, also aufzuräumen (hier kann mit `pop` gearbeitet werden). Dabei wird auch der Rückgabewert mit aufgesammelt.

## Aufgabe 2.4 (ggT mit Stack-basierter Parameterübergabe):

a) Realisieren Sie den ggT-Alrorithmus aus Aufgabe 1.3 in einer Variante entsprechend 1.4 a) mit Übergabe der Parameter über den Stack nur unter Verwendung von `push`/`pop`.

b) Realisieren Sie eine weitere Variante nach dem Muster von Aufgabe 1.4 b) unter Verwendung eines Frame-Zeigers in Register `Z`.