Projekt 1

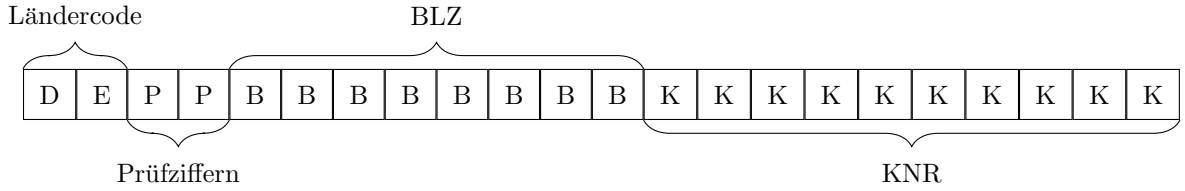
Compiler Design Lab
Prof. Dr. Sebastian Hack
Simon Moll, MSc

Vorbereitungen

Klonen Sie das Depot `https://prog2scm.cdl.uni-saarland.de/git/project1/$NAME`, wobei `$NAME` durch Ihren Benutzernamen zu ersetzen ist. Geben Sie an, dass der Klon in ein Verzeichnis namens `project1` abgelegt wird. Falls Sie *Projekt0 – Einführung in Versionsverwaltung mit git* nicht bearbeitet haben sollten, müssen Sie noch *Aufgabe 1 – Vorbereitung der git-Installation* erledigen. Außerdem möchten wir Sie drauf hinweisen, dass unser Server nur aus dem Universitätsnetz erreichbar ist. Wenn Sie von zu Hause arbeiten möchten, schauen Sie bitte in die entsprechenden Threads im Askbot.

Beachten Sie, dass Sie die Einstellungen *Assemble all files in directory* und *Initialize Program Counter to global 'main' if defined* im Menü *Settings* in MARS aktivieren müssen, um die Datei `src/main.asm` zu übersetzen, und den Programmeinstiegspunkt auf die Marke `main` zu setzen.

IBAN-Rechner (10 Punkte)



In diesem Projekt werden Sie einen IBAN-Rechner implementieren, der eine Kontonummer (KNR) und eine Bankleitzahl (BLZ) in eine deutsche IBAN nach obigem Format übersetzt, bzw. aus einer deutschen IBAN die KNR und die BLZ extrahiert. Sie finden eine allgemeine Einführung zu IBANs in der Wikipedia (`http://de.wikipedia.org/wiki/IBAN`).
Sie werden das Programm in MIPS-Assembler schreiben. Zur einfacheren Bearbeitung ist das Projekt in mehrere Aufgaben aufgeteilt. Zu jeder Aufgabe gibt es eine eigene Datei, in welche Sie das jeweilige Unterprogramm schreiben werden. Beachten Sie auch unbedingt die Hinweise am Ende der Projektbeschreibung.

1 BLZ und KNR extrahieren (2 Punkte)

Schreiben Sie ein Unterprogramm `iban2knr` (Datei `src/iban2knr.asm`), das die KNR und die BLZ aus einer deutschen IBAN extrahiert. Ihr Unterprogramm erhält folgende Argumente:

1. Ein Puffer mit einer deutschen IBAN, kodiert in ASCII mit 22 Zeichen.
2. Den Zielpuffer für die BLZ mit Platz für 8 Zeichen.
3. Den Zielpuffer für die KNR mit Platz für 10 Zeichen.

2 Rest berechnen (4 Punkte)

Schreiben Sie ein Unterprogramm, `modulo_str` (Datei `src/moduloStr.asm`), das den Divisionsrest einer Zahl berechnet, die als Zeichenkette gegeben ist. Ihr Unterprogramm erhält folgende Argumente:

1. Der Puffer mit der Zahl.
2. Die Anzahl der Ziffern im Puffer.
3. Den Teiler.

Das Ergebnis entspricht dem Rest der im Puffer dargestellten Zahl bei einer Division durch den gegebenen Teiler und wird im Rückgaberegister zurückgegeben. Dabei stellt das letzte Zeichen im Puffer an Position $Länge - 1$ die niedrigwertigste Ziffer dar. Die Zahl im Puffer ist als Dezimalzahl in ASCII-Ziffernzeichen dargestellt und hat insbesondere kein Vorzeichen. Der Teiler ist eine vorzeichenlose Zahl und immer größer als 0. Wir legen Ihnen nahe das Ergebnis ziffernweise nach dem Hornerschema zu berechnen. Beachten Sie, dass das Ergebnis auch korrekt sein muss, wenn die im Puffer dargestellte Zahl das Fassungsvermögen der Maschinenregister überschreitet. Dies können Sie durch geschickte Ausnutzung der folgenden Gleichungen erreichen ($x \bmod y$ ist der Rest der Division von x durch y):

$$(a + b) \bmod c = (a + (b \bmod c)) \bmod c$$

$$(a \times b) \bmod c = (a \times (b \bmod c)) \bmod c$$

Sie können also Zwischenergebnisse auf ihren Rest bei Division durch den Teiler reduzieren.

3 IBAN Prüfsumme berechnen (2 Punkte)

Schreiben Sie ein Unterprogramm `compute_checksum` (Datei `src/computeChecksum.asm`), das die Prüfsumme einer deutschen IBAN berechnet, die als Zeichenkette gegeben ist. Ihr Unterprogramm erhält folgendes Argument:

1. Die Startadresse eines 22 Zeichen langen Puffers mit einer deutschen IBAN.

Das Unterprogramm gibt die Prüfsumme im Rückgaberegister zurück. Orientieren Sie sich an der auf Wikipedia vorgeschlagenen Methode (`http://de.wikipedia.org/wiki/IBAN#Validierung_der_Pr.C3.BCfsumme`) zur Verifizierung der Prüfsumme. Das Unterprogramm soll den Wert zurückgeben, der in Schritt 3 der auf Wikipedia angebenen Methode zur Validierung der Prüfsumme errechnet wird. Verwenden Sie das Unterprogramm `modulo_str` aus der vorherigen Aufgabe, um die Prüfsumme zu berechnen. Sie können davon ausgeben, dass ausschließlich ASCII-Großbuchstaben und Ziffernzeichen im Puffer vorkommen.

4 IBAN erzeugen (2 Punkte)

Schreiben Sie ein Unterprogramm `knr2iban` (Datei `src/knr2iban.asm`), das aus einer KNR und einer BLZ eine deutsche IBAN mit gültigen Prüfziffern berechnet. Ihr Unterprogramm erhält folgende Argumente:

1. Der Zielpuffer für die IBAN mit Platz für 22 Zeichen.
2. Die BLZ als 8 Zeichen langer Puffer.
3. Die KNR als 10 Zeichen langer Puffer.

Greifen Sie auf das Unterprogramm `compute_checksum` aus der letzten Aufgabe zurück, um die Prüfsumme zu berechnen. Die generierte IBAN darf nur aus ASCII-Großbuchstaben und -Ziffernzeichen bestehen.

5 IBAN Prüfsumme ohne Zwischenspeicher (2 Bonuspunkte)

Schreiben Sie eine Variante von `compute_checksum` (Datei `src/bonus/computeChecksum.asm`), welche die IBAN Prüfsumme berechnet ohne Zwischenergebnisse in den Speicher zu schreiben. Konkret disqualifiziert jeglicher Syscall oder Schreibzugriff auf den Speicher die Lösung.

Tests selber ausführen und debuggen

Verwenden Sie den Befehl `./run_tests` im Hauptverzeichnis des Projekts, um ihre Implementierung testen zu lassen. Wenn ein Test, z.B. `tests/pub/test_X.asm`, fehlschlägt können Sie ihn in MARS debuggen. Der Befehl `./build_testbox tests/pub/test_X.asm` kopiert den Test zusammen mit ihrer Implementierung in den neuen Ordner `testbox/`, wo Sie ihn mit MARS starten können. Beachten Sie jedoch, dass nur die Dateien an den in der Aufgabenstellung angegebenen Pfaden in die Bewertung und die automatischen Tests eingehen. Buchen Sie also insbesondere keine Dateien in `testbox/` in das Versionsverwaltungssystem ein.

Skripte für die Bonusaufgabe

Analog zu den regulären Testprogrammen gibt es die Skripte `run_tests_bonus` und `build_testbox_bonus` mit denen Sie Ihre Lösung der Bonusaufgabe debuggen können. Das Skript `build_bonusbox` kopiert außerdem das Projekt mit der Implementierung der Bonusaufgabe in den Ordner `bonusbox/`, wo Sie es mit MARS starten können.

Abgabe Ihrer Lösung

git push bis Ende 19. Mai 2015, 23:59.

Hinweise

1. Die Datei `src/util.asm` enthält nützliche Unterprogramme, die Sie für dieses Projekt verwenden können. Einige der dort implementierten Unterprogramme sollten Ihnen aus dem MIPS-Übungsblatt bekannt sein. Diese Datei sollten Sie nicht verändern, da wir für die automatischen Tests die Originalversion verwenden.

2. Für Ihre Abgabe werden ausschließlich die Dateien im Ordner `src/` berücksichtigt, deren Namen in der Aufgabenstellung angegeben sind.

3. Beachten Sie, dass nur die auf der virtuellen Maschine installierte Version von MARS von uns unterstützt wird. Diese weicht von den offiziellen Versionen in Details ab.

4. Wir weisen aus rechtlichen Gründen darauf hin, dass keinerlei Garantien für die erzeugten IBANs, BLZs und KNRs gegeben werden.