Programmierung 2 (SS 2021)
Universität des Saarlandes
Fakultät MI
Compiler Design Lab

Projekt 1

Prof. Dr. Sebastian Hack
Julian Rosemann, B. Sc.
Thorsten Klößner, M. Sc.
Julia Wichlacz, M. Sc.
Maximilian Fickert, M. Sc.

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. Wir 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 *Discourse*.
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.s` zu übersetzen, und den Programmeinstiegspunkt auf die Marke `main` zu setzen.
Im Folgenden bezieht sich der Begriff Zeichen immer auf die 1 Byte große Kodierung eines Zeichens gemäß des ASCII Standards.

Tic Tac Toe

In diesem Projekt werden Sie das beliebte Spiel Tic-Tac-Toe in MIPS Assembler implementieren. Sie finden eine erstaunlich ausführliche Erklärung der Spielregeln bei Wikipedia
(`https://de.wikipedia.org/wiki/Tic-Tac-Toe`).
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. Die Datei `src/main.s` ruft die von Ihnen zu schreibenden Unterprogramme auf und darf nicht von Ihnen verändert werden. Beachten Sie auch unbedingt die Hinweise am Ende der Projektbeschreibung.

1 Das Spielfeld ausgeben (3 Punkte)

Schreiben Sie ein Unterprogramm `drawBoard` (Datei `src/view.s`), das das Spielfeld darstellt. Ihr Unterprogramm erhält folgende Argumente:

1. Die Adresse des Spielfeldes.

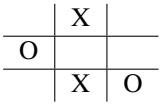
1.1 Darstellung des Spielfeldes

Die Spielerzeichen sind die ASCII-Großbuchstaben `'X'` (ASCII-Code 88) und `'O'` (ASCII-Code 79). Die vertikalen Linien werden durch das Zeichen `'|'` (ASCII-Code 124), horizontale Linien mit dem Minuszeichen `'-'` (ASCII-Code 45) Leere Spielfelder werden durch das normale Leerzeichen `' '` (ASCII-Code 32) dargestellt.

2 Einen Zug durchführen (3 Punkte)

Schreiben Sie ein Unterprogramm `takeTurn` (Datei `src/turn.s`), das ein Spielerzeichen an einer angegebenen Position setzt, falls dies möglich ist. Sie erhalten die Position kodiert in ASCII-Zeichen. Beachten Sie dabei, dass Zeilen/Spalten beginnend mit der 0 nummeriert sind (Zeile '1' ist logisch die zweite Zeile, etc.). Ihr Unterprogramm erhält folgende Argumente:

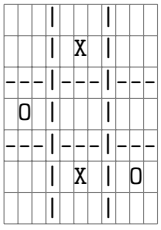
1. Die Adresse des Spielfeldes.
2. Das ASCII-Zeichen des Spielers, der am Zug ist.
3. Die Zielzeile als ASCII-Zeichen.
4. Die Zielspalte als ASCII-Zeichen.



(a) Logischer Zustand des Spielfelds.

```
   |   |
   | X |
---|---|---
 O |   |
---|---|---
   | X | O
   |   |
```

(b) Konsolenausgabe der Prozedur `drawBoard`.



(c) Zeichenposition der Konsolenausgabe.

' '	'X'	' '	'O'	' '	' '	' '	'X'	'O'

(d) Byte-weiser Speicherauszug des Spielfelds ab seiner Startadresse (Adresse des Spielfelds). Jedes Byte kodiert ein Kästchen des Spielfelds als ASCII-Code ('O', 'X' oder ' '). Die Kästchen des Spielfelds liegen zeilenweise, beginnend mit der obersten Zeile, von links nach rechts im Speicher.

Ist der angegebene Spielzug nicht möglich, gibt das Unterprogramm die Zahl 1 zurück und lässt das Spielfeld unverändert. Ist der Spielzug möglich, so schreibt das Unterprogramm das Zeichen des Spieler an die entsprechende Zeile und Spalte im Spielfeld. Im Erfolgsfall wird die Zahl 0 zurückgegeben. Ein Spielzug ist möglich, wenn folgende Bedingungen gelten:

- Die übergebenen ASCII Zeichen in Argument drei und vier beschreiben eine gültige Zielzeile und -spalte auf dem Spielfeld.

  Hinweis: Beispielsweise entspricht die Angabe Zeile '2' und Spalte '0', dem unteren linken Kästchen auf dem ausgegebenen Spielfeld. Die Angabe Zeile 'W' und Spalte '9' ist ungültig.

- Das Spielfeld ist an der angegebenen Zeile und Spalte frei (enthält das ASCII Leerzeichen ' ' an der entsprechenden Adresse).

3 Siegbedingung prüfen (4 Punkte)

Schreiben Sie ein Unterprogramm `testWin` (Datei `src/check.s`), das überprüft, ob ein bestimmter Spieler gewonnen hat. Ihr Unterprogramm erhält folgende Argumente:

1. Die Adresse des Spielfeldes.
2. Das ASCII-Zeichen des Spielers, auf dessen Sieg geprüft werden soll.

Das Unterprogramm bestimmt, ob der Spieler mit dem übergebenen Zeichen auf dem aktuellen Spielfeld gewonnen hat. Hat dieser Spieler gewonnen, so gibt das Unterprogramm den Wert 1 zurück. Ansonsten gibt das Unterprogramm 0 zurück.

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.s`, fehlschlägt, können Sie ihn in MARS debuggen.
Der Befehl `./build_testbox tests/pub/test_X.s` kopiert den Test zusammen mit ihrer Implementierung in den neuen Ordner `testbox/`, wo Sie ihn mit MARS starten können. Wenn Sie etwas an den Dateien im eigentlichem Abgabeordner ändern müssen Sie erneut den Befehl `./build_testbox tests/pub/test_X.s` eingeben, damit die Änderungen in der Testbox übernommen werden. 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.

Abgabe Ihrer Lösung

git push bis Ende 11. Mai 2021, 23:59.

Hinweise

1. Für Ihre Abgabe werden ausschließlich die Dateien im Ordner `src/` berücksichtigt, deren Namen in der Aufgabenstellung angegeben sind.
2. Halten Sie sich an die MIPS-Aufrufskonvention!
3. Sie müssen davon ausgehen, dass wir die Dateien der Teilaufgaben einzeln Testen: Sie können nicht Unterprogramme aus anderen Dateien aufrufen, die nicht explizit in der Aufgabenstellung vorgeben sind.
4. 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.
5. Dokumentieren Sie Ihr Programm mit Codekommentaren. Dadurch wird der Code lesbarer und Sie können das Programm leichter in MARS debuggen.

Viel Erfolg!