universität innsbruck

Proseminar Rechnerarchitektur

Aufgabenzettel 8

Wintersemester 2021/22

25. November 2021

Zu bearbeiten bis Donnerstag, den **2. Dezember**.

1 ARM Setup

In der Vorlesung haben Sie gelernt, welche Werkzeuge Sie zum Programmieren und Ausführen von ARM-Assemblerprogrammen benötigen: einen Texteditor, einen ARM-Assembler, einen ARM-Linker, und einen ARM-Emulator. Alle diese Programme sind auf den Computern im Rechnerraum unter Linux installiert. Alternativ können Sie sich auch per SSH auf den zid-gpl Server verbinden. (Hilfe dazu finden Sie unter `https://www.uibk.ac.at/zid/systeme/linux/lpccs_4/benutzeranleitung_zid-gpl.html`.) Sie können die Werkzeuge auch bei sich lokal installieren, wenn Sie das möchten.

Um die jeweilige Toolchain zu testen, machen Sie bitte Folgendes:
Laden Sie das Beispielprogramm `hello.S` von OLAT herunter. Assemblieren Sie den Kode wie in der Vorlesung besprochen mit dem Befehl `arm-linux-gnu-as hello.S -o hello.o`. Die Objektdatei linken Sie dann mit dem Befehl `arm-linux-gnu-ld hello.o -o hello`. Führen Sie die resultierende Datei mit `qemu-arm ./hello` aus.

Wenn `Hello Innsbruck!` ausgegeben wird, dann funktioniert Ihr Setup.

2 Taschenrechner?!

Schreiben Sie die folgenden Zuweisungen als ARM-Assemblerbefehle auf. Benutzen Sie dabei wenn möglich den Barrel-Shifter, um die notwendige Anzahl an Taktzyklen gering zu halten.

a) `r2 = r1 + 7`

b) `r6 = r1 * 32`

c) `r6 = r1 * 17`

d) `r9 = r0 - r4 : 16` (Punkt vor Strich, nur der ganzzahlige Anteil des Quotienten)

3 Immediates Laden

Welche der folgenden Befehle können mit jeweils einer gültigen `MOV` oder `MVN`-Instruktion ersetzt werden? Begründen Sie Ihre Antworten!

a) `LDR r0, =2480`

b) `LDR r0, =1240`

c) `LDR r0, =0xfa4fffff`

d) `MOV r0, #0x5b, ROR #1`

4 Sequenzielle Logik in der Praxis

Betrachten Sie das ARM-Programm auf der linken Seite. Es wurde zu Maschinencode assembliert, liegt ab Adresse `0x0` im Speicher und wird nun ausgeführt. In der Tabelle auf der rechten Seite geben wir die Belegung der Register `pc`, `r0` und `r1`, jeweils vor dem Ausführen der nächsten Instruktion. Vervollständigen Sie die Tabelle, solange bis sich eine Zeile wiederholt. Legen Sie besonderes Augenmerk auf das `pc`-Register. Sehen Sie den Zusammenhang zur sequenziellen Logik?

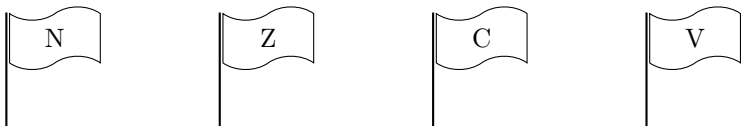
```
1  LDR r0, =0x1
2  LDR r1, =0x5
3  ADD r0, r0, r1
4  CMP r0, #0x8
5  LDRLT pc, =0x8
6  MOV pc, #0
```

pc	r0	r1
0x0	?	?
0x4	0x1	?
0x8	0x1	0x5
⋮	⋮	⋮

(Fortsetzung auf der nächsten Seite)

5 Fun with Flags

In der Vorlesung wurden Ihnen diese vier Flags vorgestellt:



negative zero carry overflow

Tragen Sie in die unten stehenden Tabellen in jeder Zeile ein, wie sich die Belegung der Flags anhand der gegebenen Befehle ändert, wenn diese von oben nach unten nacheinander abgearbeitet werden. Tragen Sie dazu ein, ob die jeweiligen Flags gesetzt (1) bzw. nicht gesetzt (0) sind, oder ob keine Aussagen darüber möglich ist (?). Eine Tabelle wurde bereits als Beispiel ausgefüllt.

Befehl		Flag			
		N	Z	C	V
		?	?	?	?
`LDR`	`r0, =25`	?	?	?	?
`LDR`	`r1, =7`	?	?	?	?
`CMP`	`r1, r0`	1	0	0	0
`SUB`	`r1, r0, r1`	1	0	0	0
`ADDS`	`r1, r0, r1`	0	0	0	0

Befehl		Flag			
		N	Z	C	V
		?	?	?	?
`LDR`	`r0, =0xf0f`				
`LDR`	`r1, =0x0f0`				
`AND`	`r2, r0, r1`				
`EORS`	`r3, r0, r0`				
`ORRS`	`r2, r0, r1`				

Befehl		Flag			
		N	Z	C	V
		?	?	?	?
`LDR`	`r0, =42`				
`LDR`	`r1, =66`				
`CMP`	`r0, r1`				
`ADD`	`r1, r0, r1`				
`SUBS`	`r0, r0, r1`				

Befehl		Flag			
		N	Z	C	V
		?	?	?	?
`LDR`	`r0, =131072`				
`LDR`	`r1, =32768`				
`ADDS`	`r2, r0, r1`				
`RSBS`	`r3, r0, r1`				
`CMN`	`r0, r1`				
`MUL`	`r4, r0, r1`				