RA FS 21 Serie 4

Abdelhak Lemkhenter, Adrian Wälchli, Sepehr Sameni

Die vierte Serie ist bis Dienstag, den 4. Mai 2021 um 15:00 Uhr zu lösen und auf ILIAS hochzuladen. Für Fragen steht im ILIAS jederzeit ein Forum zur Verfügung. Allfällige unlösbare Probleme sind uns so früh wie möglich mitzuteilen, wir werden gerne helfen. Viel Spass!

Theorieteil

Gesamtpunktzahl: 13 Punkte

1 Sign Extension (1 Punkt)

Erläutern Sie, weshalb die Sign-Extension gerade bei Branch-, Load- und Store-Befehlen gebraucht wird.

2 Logische und bitweise Operationen (1 Punkt)

Bestimmen Sie die jeweilige Ausgabe:

Operation	Resultat
`0x0 && 0xEF`	
`0xD3 & 0x5B`	
`0x0 || 0xEF`	
`0xA3 | 0x3A`	
`!0xFE`	
`~0xFE`	

3 Unendliche Schleife (2 points)

Ein Programmierer hatte die Absicht, dass das folgende MIPS Programm den Teil `foo` in Zeile 8 exakt 9 mal ausführt. Leider terminiert das Programm aber nicht. Erklären Sie warum und beheben Sie das Problem ohne die Programmlogik substantiell abzuändern (bestehende Instruktionen umordnen, modifizieren oder entfernen ist nicht erlaubt).

```
1  li $t0, 9
2
3  loop:
4  jal foo
5  addi $t0, $t0, -1
6  bne $t0, 0, loop
7
8  foo:
9  li $t1, 0xAFFFFF04
10 sw $t0, 0($t1)
11 jr $ra
```

4 `bne` statt `beq` (2 Punkte)

Welche Änderungen sind bei der in der Vorlesung vorgestellten MIPS-Implementation (siehe "Basic MIPS Architecture Review") notwendig, um `bne` statt `beq` zu implementieren.

(a) Für die Singlecycle-Implementation

(b) Für die Multicycle-Implementation

5 Pipeline Register (1 Punkt)

Wozu werden die Register (siehe Folie 6, "Basic MIPS Pipelining Review") zwischen den einzelnen Berechnungsstufen benötigt?

6 Pipelining Hazard (2 Punkte)

Erklären Sie den Unterschied zwischen Control-, Data- und Structural-Hazards.

7 Stall (2 Punkte)

Erläutern Sie, warum es auf Folie 15, im Gegensatz zur Folie 19, genügt, nur zwei Taktzyklen zu warten. (Die Seitenangaben beziehen sich auf das Kapitel "Basic MIPS Pipelining Review")

8 Data Hazard (2 Punkte)

Geben Sie sämtliche Data Hazards im folgenden Code an. Bei welchen Abhängigkeiten handelt es sich um Data Hazards, die mittels Forwarding behoben werden können? Bei welchen Abhängigkeiten handelt es sich um Data Hazards, die zu einem *stall* führen?

```
1 add $t0, $t5, $t4
2 lw $s2, 0($t0)
3 sub $s3, $t0 ,$s2
4 sw $t4, 4($s3)
```

Programmierteil

Die Programmieraufgaben können in Zweiergruppen gelöst werden. Sie und Ihre Gruppenmitglieder arbeiten gemeinsam am Code oder teilen sich Unteraufgaben auf. Um sicherzustellen, dass jede Programmiererin und jeder Programmierer den gesamten Code versteht und erklären kann, verlangen wir von jedem Mitglied eine leicht unterschiedliche Abgabe. Wie unten beschrieben unterscheiden sich die Versionen nur um eine spezielle Funktionalität für die die einzelne Teilnehmerin oder der einzelne Teilnehmer selbst verantwortlich ist.

Vorbereitung

(a) Laden Sie das Codeskelett, das MARS MIPS Tutorial und die JAR Datei von ILIAS herunter.

(b) Folgen Sie dem Tutorial und machen Sie sich mit dem Simulator vertraut.

(c) Wir werden in dieser Übung das *Digital Lab Sim* Tool verwenden. Dies können Sie im Menu unter `Tools` → `Digital Lab Sim` öffnen und durch Klicken auf *Connect to MIPS* aktivieren.

Countdown Clock

Sie werden in dieser Übung mit dem MARS MIPS Simulator eine Countdown-Uhr programmieren. In Abbildung 1a sehen Sie eine Vorschau dieser Uhr. Folgende Funktionalität soll vorhanden sein:

- Sie besteht aus zwei Zifferen auf denen eine maximale Zeit von 99 Sekunden angezeigt werden kann.
- Die Uhr aktualisiert sich jede Sekunde und zählt herunter.
- Die Ziffern beginnen zu blinken sobald die Zeit abgelaufen ist.

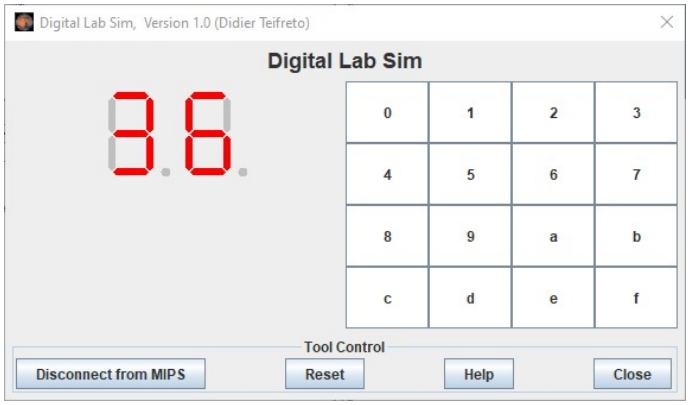
Aufgaben

(a) Studieren Sie den bereits vorhandenen Code. Die relevanten Stellen für die Unteraufgaben sind mit Kommentaren markiert. Tutorials und Beispielmaterial zur Verwendung des MARS Simulators ist in ILIAS verfügbar.

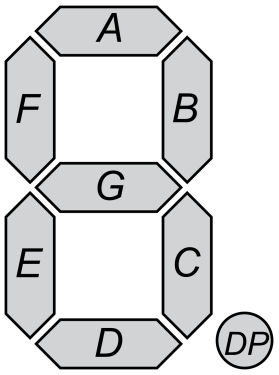
(b) Ergänzen Sie den Kopf des Quellcodes mit ihren Namen.

(c) Vervollständigen Sie die Tabelle 1 mit den korrekten Werten zum ansteuern der 7-Segment Anzeige und übertragen Sie die Hexadezimalwerte in den Quellcode unter Abschnitt (c).

(d) Implementieren Sie die Subroutine `write_digit`. Testen Sie diese indem Sie sie im Abschnitt `main` mit Testwerten aufrufen. Stellen Sie sicher, dass die Anzeige für alle Ziffern korrekt ist bevor Sie fortfahren. Die Kommentare im Quellcode enthalten Tipps.



(a) Das *Digital Lab Sim* GUI ist Teil von MARS (öffnen Sie es in Tools → Digital Lab Sim).



(b) Verwenden Sie diese Legende um die Tabelle zu ergänzen.

Abbildung 1: Eine Countdown-Uhr mit zwei 7-Segment Anzeigen.

(e) Implementieren Sie die Hauptschleife für den Countdown. Die Kommentare im Quellcode enthalten bereits wertvolle Tipps. Zudem stellen wir eine Subroutine zur Verfügung, die eine zweistellige Ganzzahl in ihre Ziffern aufteilt.

(f) Erweitern Sie Ihr Programm mit einer Animation die am Ende des Countdowns startet.

Student 1: In dieser Version sollen die Ziffern blinken.
Student 2: In dieser Version sollen die Dezimalpunkte blinken.

(g) Studieren Sie die Subroutine `get_digits`. Für Instruktionen die Sie nicht kennen können Sie die MIPS Referenz herbeiziehen (verfügbar in ILIAS). Ergänzen Sie jede Codezeile von `get_digits` mit einem kurzen Kommentar zur Erklärung.

(h) Laden Sie Ihren Quellcode in ILIAS bis Dienstag, den 4. Mai 2021 um 15:00 Uhr hoch. Wenn Sie in einer Gruppe arbeiten müssen Sie zwei Versionen hochladen, eine pro Student/in (Sie können beide Versionen in den gleichen Gruppenordner hochladen).

Digit	DP	G	F	E	D	C	B	A	Hex
0	0	0	1	1	1	1	1	1	3F
1	0								
2	0								
3	0								
4	0								
5	0								
6	0								
7	0								
8	0								
9	0								

Tabelle 1: Ermitteln Sie die Hexadezimalcodes für jede Ziffer. Diese benötigen Sie im Codeteil (c). Das tiefstwertigste Bit entspricht dem Segment A und das höchstwertigste Bit entspricht dem Dezimalpunkt (DP).