

10.5 / 20

# Minitest 1

Minh Khue Pham (2579036)

04.05.2021 13:55:20 - 14:10:13

## g Multiple Choice (10 Punkte)

Entscheiden Sie ob die folgenden Aussagen wahr oder falsch sind. **(10 Punkte)**  
Für korrekte Antworten bekommen Sie 1 Punkt. Für keine Antwort bekommen Sie 0.5 Punkte. Für eine falsche Antwort gibt es 0 Punkte.

1. Nach Ausführung des Befehls `li $0 1` liegt der Wert 1 in Register \$0

Response:

- ☐ wahr
- ☒ falsch

2. Ein ASCII-Zeichen lässt sich in 1 Byte kodieren.

Response:

- ☒ wahr
- ☐ falsch

3. Der Befehl `beqz $0 0` erzeugt eine Endlosschleife.

Response:

- ☐ wahr
- ☒ falsch

4. Die Assemblerdirektive `.data` markiert den Beginn des Codesegments.

Response:

- ☐ wahr
- ☒ falsch

5. Um ein Wort zu laden muss die Speicheradresse durch 4 teilbar sein.

**Response:**

- ☒ wahr
- ☐ falsch

6. Nach MIPS-Aufrufkonvention muss die aufgerufene Funktion alle s-Register vor Rückprung wiederherstellen.

**Response:**

- ☒ wahr
- ☐ falsch

7. Ein Befehlswort des MIPS-Prozessors ist 32-bit lang.

**Response:**

- ☒ wahr
- ☐ falsch

8. Sprungbefehle die mit j beginnen verwenden relative Adressierung.

**Response:**

- ☐ wahr
- ☒ falsch

9. Die Elemente einer Reihung liegen fortlaufend im Speicher.

**Response:**

- ☒ wahr
- ☐ falsch

10. Dynamisch allozierter Speicher wird immer auf dem Keller angefordert.

**Response:**

- ☒ wahr
- ☐ falsch

1,5

**Ausführungsprotokolle (5 Punkte)**

Sehen Sie sich folgendes MIPS Programm an:

Befehlsadresse	Befehl
0x00400000	li \$a0 5
0x00400004	li \$a1 4
0x00400008	bgt \$a1 \$a0 2
0x0040000c	subu \$a0 \$a0 \$a1
0x00400010	b -3
0x00400014	li \$v0 1
0x00400018	beq \$a0 \$zero 1
0x0040001c	and \$v0 \$v0 \$zero

Erstellen Sie ein Ausführungsprotokoll für das oben beschriebene Programm indem Sie die unten stehende Tabelle ausfüllen. **(5 Punkte)**

Im ersten Schritt befindet sich der Programmzähler an der Stelle 0x00400000 und die Register \$a0, \$a1 und \$v0 haben einen unbestimmten Wert. Sie dürfen aufhören, sobald der Programmzähler auf eine Adresse zeigt, an der keine der gelisteten Instruktionen zu finden ist.

*Hinweis: Das Programm benötigt weniger als 15 Schritte.*

1. \$pc = 0x00400000    \$a0 = ?    \$a1 = ?    \$v0 = ?

2.     ✓

3.     ✓

4.     4 > 5 ?

5.     (✓)

6.

7.	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
8.	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
9.	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
10.	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
11.	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
12.	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
13.	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
14.	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
15.	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

### Reihungen (5 Punkte)

Schreiben Sie ein MIPS-Unterprogramm, das alle **negativen** Zahlen in einer Reihung mit 0 überschreibt. **(5 Punkte)**

Dem Unterprogramm wird die Adresse des ersten Elementes der Reihung in Register \$a0 und die Adresse direkt hinter dem letzten Element in Register \$a1 übergeben. Die Elemente der Reihung sind 4 Byte groß. Beenden Sie das Programm am Ende des Unterprogramms ordnungsgemäß mit einem syscall (anstatt vom Unterprogrammaufruf zurückzukehren).

Hinweise:

- Befindet sich in Register \$v0 eine 10, so wird bei einem syscall das Programm beendet.
- Sie müssen nicht die Aufrufkonvention implementieren.

#### Response:

```
.text
bltz $a0 0
move $a0
```

## MIPS-Befehlsreferenz

Art	Mnemonic	Argumente	Bedeutung	Kommentar
Arithmetik	addu	$\$d \ \$s \ \$t$	$\$d \leftarrow \$s + \$t$	Addiere zwei Registerinhalte
	addiu	$\$d \ \$s \ i$	$\$d \leftarrow \$s + si$	Addiere Registerinhalt zu Immediate
	subu	$\$d \ \$s \ \$t$	$\$d \leftarrow \$s - \$t$	Subtrahiere zwei Registerinhalte
	mul	$\$d \ \$s \ \$t$	$\$d \leftarrow \$s \cdot \$t$	Multipliziere zwei Registerinhalte
	div	$\$d \ \$s \ \$t$	$\$d \leftarrow \$s / \$t$	Dividiere zwei Registerinhalte
	rem	$\$d \ \$s \ \$t$	$\$d \leftarrow \$s \% \$t$	Rest bei der Division
Bitoperationen	and	$\$d \ \$s \ \$t$	$\$d \leftarrow \$s \& \$t$	Bitweises Und
	andi	$\$d \ \$s \ i$	$\$d \leftarrow \$s \& zi$	Bitweises Und mit Immediate
	or	$\$d \ \$s \ \$t$	$\$d \leftarrow \$s   \$t$	Bitweises Oder
	ori	$\$d \ \$s \ i$	$\$d \leftarrow \$s   zi$	Bitweises Oder mit Immediate
	xor	$\$d \ \$s \ \$t$	$\$d \leftarrow \$s \wedge \$t$	Bitweises exkl. Oder
	xori	$\$d \ \$s \ i$	$\$d \leftarrow \$s \wedge zi$	Bitweises exkl. Oder mit Immediate
	nor	$\$d \ \$s \ \$t$	$\$d \leftarrow \overline{\$s   \$t}$	Nicht-Oder, NOR
	lui	$\$d \ i$	$\$d \leftarrow i_{15} \dots i_0 \cdot 0^{16}$	Lade obere 16 Bits eines Registers
Shifts	sll	$\$d \ \$s \ n$	$\$d \leftarrow \$s_{31-n} \dots \$s_0 \cdot 0^n$	Shifte nach links um $n$
	srl	$\$d \ \$s \ n$	$\$d \leftarrow 0^n \cdot \$s_{31} \dots \$s_n$	Shifte vorzeichenlos nach rechts um $n$
	sra	$\$d \ \$s \ n$	$\$d \leftarrow (\$s_{31})^n \cdot \$s_{31} \dots \$s_n$	Shifte vorzeichenbeh. nach rechts um $n$
	sllv	$\$d \ \$s \ \$t$	$\$d \leftarrow \$s_{31-n} \dots \$s_0 \cdot 0^n$	$n := \$t$ (s.o.)
	srlv	$\$d \ \$s \ \$t$	$\$d \leftarrow 0^n \cdot \$s_{31} \dots \$s_n$	$n := \$t$ (s.o.)
	srav	$\$d \ \$s \ \$t$	$\$d \leftarrow (\$s_{31})^n \cdot \$s_{31} \dots \$s_n$	$n := \$t$ (s.o.)
Vergleiche	slt	$\$d \ \$s \ \$t$	$\$d \leftarrow \$s \lessdot \$t ? 1 : 0$	Vorzeichenbeh. Vergl. zweier Registerinhalte
	sltu	$\$d \ \$s \ \$t$	$\$d \leftarrow \$s \lessgtr \$t ? 1 : 0$	Vorzeichenloser Vergl. zwei Registerinhalte
	slti	$\$d \ \$s \ i$	$\$d \leftarrow \$s \lessdot si ? 1 : 0$	Vorzeichenbeh. Verg. von Reg mit Immediate
	sltiu	$\$d \ \$s \ i$	$\$d \leftarrow \$s \lessgtr si ? 1 : 0$	Vorzeichenloser Verg. von Reg mit Immediate
Laden	lw	$\$d \ i(\$s)$	$\$d \leftarrow W[\$s + si]$	Lade Wort
	lh	$\$d \ i(\$s)$	$\$d \leftarrow \text{sext}_{16}^{32}(H[\$s + si])$	Lade zwei Bytes mit Vorzeichenerw.
	lb	$\$d \ i(\$s)$	$\$d \leftarrow \text{sext}_8^{32}(B[\$s + si])$	Lade Byte mit Vorzeichenerw.
	lhu	$\$d \ i(\$s)$	$\$d \leftarrow \text{zext}_{16}^{32}(H[\$s + si])$	Lade zwei Bytes ohne Vorzeichenerw.
	lbu	$\$d \ i(\$s)$	$\$d \leftarrow \text{zext}_8^{32}(B[\$s + si])$	Lade Bytes ohne Vorzeichenerw.
Speichern	sw	$\$d \ i(\$s)$	$W[\$s + si] \leftarrow \$d$	Speichere Wort
	sh	$\$d \ i(\$s)$	$H[\$s + si] \leftarrow \$d[15 : 0]$	Speichere zwei Bytes
	sb	$\$d \ i(\$s)$	$B[\$s + si] \leftarrow \$d[7 : 0]$	Speichere Byte
Sprung	beq	$\$s \ \$t \ i$	$pc \leftarrow pc + 4 \cdot (1 + \$s = \$t ? si : 0)$	Unterprogrammaufruf Indirekter Sprung Rufe Betriebssystem
	bne	$\$s \ \$t \ i$	$pc \leftarrow pc + 4 \cdot (1 + \$s \neq \$t ? si : 0)$	
	blez	$\$t \ i$	$pc \leftarrow pc + 4 \cdot (1 + \$t \leq 0 ? si : 0)$	
	bgtz	$\$t \ i$	$pc \leftarrow pc + 4 \cdot (1 + \$t > 0 ? si : 0)$	
	bltz	$\$t \ i$	$pc \leftarrow pc + 4 \cdot (1 + \$t < 0 ? si : 0)$	
	bgez	$\$t \ i$	$pc \leftarrow pc + 4 \cdot (1 + \$t \geq 0 ? si : 0)$	
	jal	$addr$	$\$ra \leftarrow pc + 4, pc \leftarrow addr$	
	jr	$\$s$	$pc \leftarrow \$s$	
	syscall			
Pseudo	li	$\$d \ i$	$\$d \leftarrow i$	Lade Konstante in Register
	la	$\$d \ l$	$\$d \leftarrow addr$	Lädt Adresse der Marke $l$
	move	$\$d \ \$s$	$\$d \leftarrow \$s$	Kopiere Registerinhalt
	not	$\$d \ \$s$	$\$d \leftarrow \overline{\$s}$	Bitweise Negation
	neg	$\$d \ \$s$	$\$d \leftarrow -\$s$	Vorzeichen ändern
	b	$i$	$pc \leftarrow pc + 4 \cdot (1 + si)$	Unbedingter Sprung
	bcc	$\$s \ \$t \ i$	$pc \leftarrow pc + 4 \cdot (1 + \$s \ cc \ \$t ? si : 0)$	$cc \in \{lt, gt, le, ge\}$ (vorzeichenbehaftet)