Hochschule Konstanz Technik, Wirtschaft und Gestaltung (HTWG)
Fakultät Informatik
Robotik und Künstliche Intelligenz
Prof. Dr.-Ing. Michael Blaich

Übung zur Vorlesung Rechnerarchitektur AIN 2

Aufgabe 1: Assembler und Maschinensprache

Die Abgabe erfolgt durch Hochladen der Lösung in Moodle. Zusätzlich wird die Lösung in der Übung nach dem Abgabetermin stichprobenartig kontrolliert.

Bearbeitung in Zweier-Teams

Team-Mitglied 1: Lukas Linde
Team-Mitglied 2: Marc Bohner

1) Assembler Instruktionen

Die folgenden Tabellen enthalten eine Reihe von Instruktionen, die Sie nacheinander für die ebenfalls in den Tabellen gegebenen Register- und Speicherinhalte ausführen sollen. Tragen Sie die Veränderungen der gelisteten Register- und Speicherinhalte jeweils in den freien Feldern der Tabellen ein.

Hinweise:

- Punkte pro Instruktion wie in der ersten Spalte der Tabelle angegeben.

		Register (Inhalte als Signed Integer)						
		$s0	$s1	$s2	$t0	$t1	$t2	$sp
P	**Instruktionen**	4	-13	-2	16	12	42	0x7FFF AF18
0,5	add $t0,$t0,$t0				32			
0,5	slti $s1,$s1,-7		1					
1	andi $s1,$sp,255		24					
1,5	lbu $t0,-12($sp)				128			
1	sw $s2,-8($t1)			-2		12		
0,5	srav $s0,$s0,$s0	0						

Speicherausschnitt					
Adresse (hexadezimal)	**Inhalt (unsigned Bytes)**		**Adresse (hexadezimal)**	**Inhalt (unsigned Bytes)**	
…		Änderung	…	…	Änderung
0x0000 000B	255		0x7FFF AF0F	255	
0x0000 000A	255		0x7FFF AF0E	255	
0x0000 0009	4		0x7FFF AF0D	255	
0x0000 0008	49		0x7FFF AF0C	128	
0x0000 0007	255		0x7FFF AF0B	0	
0x0000 0006	255		0x7FFF AF0A	0	
0x0000 0005	251		0x7FFF AF09	0	
0x0000 0004	255	254	0x7FFF AF08	255	
0x0000 0003	0		0x7FFF AF07	255	
0x0000 0002	6		0x7FFF AF06	192	
0x0000 0001	0		0x7FFF AF05	128	
0x0000 0000	5		0x7FFF AF04	48	

2) Maschinensprache

Im Folgenden ist ein Stück Programm-Code sowohl in Assemblersprache als auch in Maschinensprache gegeben. Beide Programm-Codes weisen Lücken auf. Ergänzen Sie diese Lücken.

Speicher-adresse	**Maschinenformat**								**Assembler**	**P**
1008	1010	1111	1011	0011	1111	1111	1000	0000	L1: sw $s3, -128 ($sp)	1
1012	0000	1000	0000	0000	0000	0001	0000	0000	L2: j L5	1,5
1016	0011	1001	0010	1000	0000	0000	1000	0000	L3: xori $t0, $t1, 128	1
1020	0000	0000	0001	0000	1000	0000	1100	0011	L4: sra $s0, $s0, 3	2
1024	0001	0110	0000	0000	0000	0000	1111	1111	L5: bne $s0,$zero,L4	1
1028	0000	0000	1001	0000	0001	0000	0010	0111	L6: nor $v0, $a0, $s0	1,5

3) Assemblerprogrammierung

In dieser Aufgabe implementieren Sie ihre ersten Zeilen Assemblercode. Versuchen Sie zunächst, den Code auf Papier aufzuschreiben und überprüfen Sie den Code dann im Mars-Simulation.

a) Erster Assembler Code

```
c=abs(a-b)
```

Verwenden Sie die Register `$s0`, `$s1` und `$s2` für die Variablen `a`, `b` und `c`. Die Funktion `abs(x)` berechnet den Betrag von `x`.

b) Erste Schleife

Implementieren den folgenden C Code in Assembler:

```
int a,b,c,n
a=0;
b=1;
n=10;
while n>0 {
   c=a+b;
   a=b;
   b=c;
   n=n-1;
}
```

Verwenden Sie für die Variablen `a`, `b`, `c` und `n` die Register `$s0` bis `$s3`.

7

a)
```
        sub $s2, $s0, $s1
        slt $t0, $s2, $zero
        bne $t0, $zero, Else
        j Exit
Else:   sub $s2, $zero, $s2
Exit:
```

b)
```
        add $s0, $zero, $zero
        addi $s1, $zero, 1
        addi $s3, $zero, 10
loop:   beq $s3, $zero, continue
        slt $t0, $s3, $zero
        bne $t0, $zero, continue
        add $s2, $s0, $s1
        add $s0, $zero, $s1
        add $s1, $zero, $s2
        addi $s3, $s3, -1
        j loop
continue:
```