CA-3

Lukas Ingold 20-123-998

RA FS 21 Serie 2

Abdelhak Lemkhenter, Adrian Wälchli, Sepehr Sameni

Die zweite Serie ist bis Dienstag, den 30. März 2021 um 15:00 Uhr zu lösen und (wenn möglich als .zip Datei) nur 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: 11 Punkte

1 Function Pointer (1 Punkt)

Was gibt folgendes Programmstück aus:

```
1  void callA(int a) {
2    printf("A: %i\n", a);
3  }
4
5  void callB(int a) {
6    printf("B: %i\n", a);
7  }
8
9  void callC(int a) {
10   printf("C: %i\n", a);
11 }
12
13 void (*functionPointer[3])(int) = { &callA, &callB, &callC };
14
15 functionPointer[0](20);
16 functionPointer[1](21);
17 functionPointer[2](22);
```

A: 20
B: 21
C: 22

2 Const (2 Punkte)

Beschreiben Sie die unterschiedlichen Eigenschaften der folgenden vier Deklarationen.

```
1 int * a;
2 int const * b;
3 int * const c;
4 int const * const d;
```

1: deklariert das a einen Pointer auf einen int ist
2: b ist ein pointer auf einen Konstanten int
3: c ist ein Konstanter Pointer auf einen int
4: d ist ein konstanter pointer auf einen konstanten int

3 Arrays (1 Punkt)

Was ist das Problem bei folgendem Programmstück:

```
1 int array[11] = {1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1};
2 int i;
3 for (i=0; i<=11; ++i) {
4   printf("%i ", array[i]);
5 }
```

Dieses Programm gibt den array zurück und dann den an der nächsten adresse gespeicherten wert da i immer gerade inkrementiert wird also wird i<=11 erfüllt sein und dann array[i] ausgegeben. Korrekt wäre zeile 3: for(i=0; i<=11;i++){

4 MIPS Branch Instructions (2 Punkte)

Nehmen Sie an, Register `$s2` enthalte den Wert 4, `$s1` enthalte den Wert 1 und `$s3` enthalte den Wert 20. Beschreiben Sie mit äquivalentem C-Code was folgendes Beispiel bewirkt:

```
L1: #do something
add $s2, $s2, $s1
beq $s2, $s3, L2
j L1
L2:
```

```
#include<stdio.h>

int main (){
        Int s2 = 4
        Int s1 = 1
        Int s3 = 20

        Int s2 = s2 +s1

        If (s2==s3){
            //do something
        }
}
```

5 MIPS More Branch Instructions (1 Punkt)

Wie wird der folgende Pseudo-Befehl vom Assembler erweitert?

```
bge $s2, $s3, Label
```

Springe nach Label, wenn $s2>=$s3
Slt $at, $s2, $s3
Beq $at, $zero, $Label

6 Endianness (1 Punkt)

Angenommen, ein 32-Bit `integer` werde als `word` an der Adresse 10001 abgespeichert:

Address	Stored binary value
10001	0101 1010
10002	1011 0110
10003	0101 1110
10004	1001 1010
10005	0110 1001

Welchen Wert hat die Zahl (Dezimal) und an welcher Adresse ist das *least significant byte* abgespeichert, wenn

(a) Big Endian A) 187301559052489, 10005
(b) Little Endian B) 453561464410, 10001

als Byte-Reihenfolge verwendet wird?

7 Array-Zugriff (2 Punkte)

Schreiben sie folgendes C-Programmstück in Assembler um:

```
1 void mul(short x[], int index, int mul) {
2   x[index] *= mul;
3 }
```

Nehmen Sie dabei an, dass die Adresse des ersten Arrayelements im Register `$t2`, `index` in `$t1` und `mul` in `$t4` gespeichert sind.

```
ADD $t3, $t1, $t1
ADD $t0, $t2, $t3
MULT $t0, $t4
MFLO $t0
JR $ra
```

8 MIPS Register/Hauptspeicher (1 Punkt)

Angenommen, B sei ein Array mit elf Daten-Wörtern (im Hauptspeicher), die Basisadresse des Arrays befinde sich in `$s1`. Laden Sie das letzte Wort von B mit genau einem Befehl in das Register `$s2`.

lw $s2, 44($s1) Angenommen B ist Integer Array (4 Byte) => 11*4=44

Optionale Fragen

Die folgenden Fragen beziehen sich auf die Dateien aus dem Programmierteil, sie müssen nicht beantwortet werden, helfen aber beim Verständnis des Programmierteils.

Byte-Reihenfolge

Welche Endianness (Byte-Reihenfolge) verwendet unsere MIPS-Simulation (Big-Endian oder Little-Endian)? Begründen Sie Ihre Antwort.

Deklarationen

Was wird hier deklariert? Wozu wird dies später verwendet? (in `mips.h`)

```
210 extern Operation operations[OPERATION_COUNT];
211 extern Function functions[FUNCTION_COUNT];
```

Sign Extension

Beschreiben Sie, was die Funktion `word signExtend(halfword value)` bezweckt (in `mips.c`).

Tests

Beschreiben Sie die Tests, die die bereits implementiert sind (z.B. `void test_addi()` in `test.c`).

Programmierteil

Ihre Aufgabe ist es, das gegebene Programmgerüst wie folgt zu vervollständigen:

(a) Laden Sie die zu Beginn erwähnten Dateien von Ilias herunter und studieren diese aufmerksam. Versuchen Sie zu verstehen, was die bereits vorhandenen Teile bedeuten, die optionalen Fragen aus dem vorherigen Abschnitt können Ihnen dabei behilflich sein.

(b) Tragen Sie Ihren Namen sowie den Namen einer allfälligen Übungspartnerin oder eines allfälligen Übungspartners an den vorgesehenen Stelle in den Dateien `mips.c` und `test.c` ein.

(c) Implementieren Sie die Funktion `storeWord` (in `mips.c`).

(d) Schreiben Sie sinnvolle und ausführliche Tests für folgende MIPS-Operationen (in `test.c`).

`lw`, `ori` und `sub`

(e) Implementieren Sie die folgenden MIPS-Operation gemäss den Spezifikationen im Buch "Rechnerorganisation und -entwurf" von D.A. Patterson und J.L. Hennessy (in `mips.c`).

`add`, `addi`, `jal`, `lui` und `sw`

Sie finden die Spezifikationen auch im PDF "MIPS Reference Data", das Sie von ILIAS herunterladen können (Literatur.zip)

(f) Stellen Sie sicher, dass Ihre Implementation ohne Fehler und Warnungen kompilierbar ist, überprüfen Sie dies mit `make`

(g) Stellen Sie sicher, dass Ihre Implementation die gegebenen und Ihre eigenen Tests ohne Fehler und Warnungen absolviert, überprüfen Sie dies mit `make test`

(h) Erstellen Sie aus Ihrer Lösung eine Zip-Datei namens `<nachname>.zip` (wobei `<nachname>` natürlich durch Ihren Nachnamen zu ersetzen ist).

(i) Geben Sie die Datei elektronisch durch Hochladen in Ilias ab.