Rechnerarchitektur Serie 1

Dominik Bodenmann 08-103-053
Orlando Signer 12-119-715

8. März 2014

1 Theorieteil

1.1 Aufgabe 1

5 Bytes, 4 Bytes für die Zeichen (T,e,s,t) und 1 Byte für \0.

1.2 Aufgabe 2

Listing 1: int-Array

```
1 int a[10];
2
3 int getAt(int i) {
4   return a[i];
5 }
6
7 int getAtWithPointer(int *a, int i) {
8   return *(a+i);
9 }
```

Listing 2: short-Array

```
1 short a[10];
2
3 short getAt(int i) {
4   return a[i];
5 }
6
7 short getAtWithPointer(short *a, int i) {
8   return *(a+i);
9 }
```

Bei Pointern beziehen sich die Rechenoperationen immer auf die Breite des Variablentyps (short 2 Byte, int 4 Byte). Somit zeigt auch *(a+i) auf die i-te Stelle im Array, egal ob es sich um ein short- oder int-array handelt.

1.3 Aufgabe 3

Listing 3: Ausgabe

```
1 bffff844
2 3ade68b1
3 68
4 de
5 bffff847
```

1. Der Wert von p: Die erste Speicheradresse von b.
2. p wird als long (4 Bytes) dereferenziert und danach inkrementiert. Da p vom Typ void ist, wird er nur um 1 erhöht.
3. p wird als char (1 Byte) dereferenziert und danach inkrementiert.
4. p wird als unsigned char (1 Byte) dereferenziert und danach inkrementiert.
5. Der Wert von p: Die 4. Speicheradresse von b.

1.4 Aufgabe 4

Preincrement: i = 1338, j = 1338
Postincrement: i = 1338, j = 1337

1.5 Aufgabe 5

1. Codestück

- 3. Zeile: x und px werden dereferenziert, es zeigen beide auf die 0. Stelle im Array. Ausgabe: "1 1"
- 5. Zeile: px wurde inkrementiert, zeigt jetzt auf die 1. Stelle. Ausgabe: "1 2"

2. Codestück

```
*(py--) = 10;
```

setzt zuerst y auf 10, danach wird py dekrementiert und auf 11 gesetzt. Die Ausgabe ist also "10 11"

1.6 Aufgabe 6

Grösse des structs: 4 * 1 Byte + 2 * 1 Byte + 2 Byte = 8 Bytes
Grösse des union: max(8 * 1 Byte, 4 Byte, 2 Byte) = 8 Bytes