

RA FS 21    Serie 1

Abdelhak Lemkhenter, Adrian Wälchli, Sepehr Sameni

Die erste Serie ist bis Dienstag, den 16. März 2021 um 15:00 Uhr zu lösen. Ihre Abgabe in ILIAS besteht aus zwei Teilen: Ein PDF mit den Lösungen zu den theoretischen Aufgaben und eine Zip-Datei mit allen Quellcode-dateien des Programmier-teils. 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: 12 Punkte

1    String (1 Punkt)

Welchen Inhalt hat der Speicherblock der Variable *word* im folgenden Beispiel?

```
1 char word[7]="string";        string\0
```

2    Array-Zugriff mit Pointern (1 Punkt)

Gegeben ist folgende `setValueAt` Funktion, welche ein bestimmtes Element in einem `double`-Array an gegebener Position aktualisiert. Geben Sie eine äquivalente Implementation mittels direkter Pointer Arithmetik an. Identifizieren und **beheben** Sie potenzielle Probleme im bereitgestellten Code.

```
1 int setValueAt(double *x, int i, double value) {        int setValueAt(double *x, int i, double value){
2        x[i] = value;                                        x=&x[i];
3        return 0;                                            *x= ( int ) value;
4        }                                                     return 0;
                                                              }
```

3    Pointer-Typen (2 Punkte)

Gegeben sei der folgende Code:

```
long a = 1234567890; /* Hex: 499602d2 */
long b = 1000000000; /* Hex: 3b9aca00 */
```

Das führt zu folgendem Speicherinhalt (Ausschnitt):

Adresse    Inhalt (Hex)

```
...
bffff604    00
bffff605    ca
bffff606    9a
bffff607    3b
bffff608    d2
bffff609    02
bffff60a    96
bffff60b    49
...
```

Hinweis: Byte-Reihenfolge ist Little-Endian. Das heisst das niedrigstwertige Byte steht an der tiefsten Speicher-Adresse, die weiteren Bytes folgen in aufsteigender Wertigkeit.

Geben Sie an, was der folgende Code ausgibt und erklären Sie warum (Sie können annehmen, dass es sich um eine 32-bit Architektur handelt, i.e. `sizeof(int) = sizeof(long) = sizeof(void*)`, und dass char vorzeichenbehaftet ist):

```
1 void * p = &b;                                                2: bffff607
2 printf("%x\n", p);                                            3: 3b9aca00
3 printf("%x\n", *(long*)p++);                                4: ffffffff
4 printf("%x\n", *(char*)p++);                                5: 9a
5 printf("%x\n", *(unsigned char*)p++);                      6: bffff60e
6 printf("%x\n", p);
```

4    Parameterübergabe (2 Punkte)

Gegeben sei das folgende Programm. Geben Sie an, welchen Wert die Variablen `i` und `j` nach Programmdurchlauf haben.

```
1 int main () {                                                i:6
2 int i, j;                                                    j:6
3 i = 5;
4 j = increment(&i);
5 }
6
7 int increment(int *x) {
8 return ++(*x);
9 }
```

Welchen Wert hätten die Variablen `i` und `j`, wenn die Funktion `increment` wie folgt definiert wäre?

```
*7 int increment(int *x) {                                    i:6
*8 return (*x)++;                                            j:5
*9 }
```

5    Pointer Arithmetik (2 Punkte)

Beschreiben Sie, was bei jedem `printf` im untenstehenden Programmstück ausgegeben wird.

```
1 short x[4] = {1, 2, 3, 4};
2 int *px = x;                                                3: 1 1
3 printf("%i %i\n", *x, *(short *)px);                      5: 1 2
4 px++;
5 printf("%i %i\n", *x, *(short *)px);
```

Beschreiben Sie, welche Ausgabe das folgende Programmstück zeigt. Welche Probleme können auftreten?

```
1 short x = 3;                                                20 21
2 short *px = &x;                                            Memory wird überschrieben bei px--
3 *(px--) = 20;                                              Bsp wird dann 20 bei adresse 1002 gespeichert und dann dekrementiert und so wären wir dann bsp bei
4 *px = 21;                                                    adresse 1001 dort wird 21 gespeichert
5 printf("%i %i\n", x, *px);
```

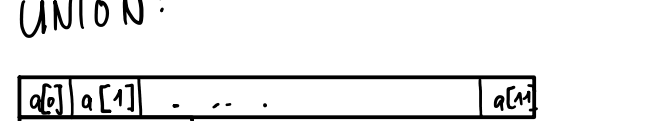
6    Structs und Unions (3 Punkte)

Erklären Sie, welchen Wert dieser Programmteil ausgibt:

```
1 struct {
2 char a[10];
3 char b;
4 char c;
5 short int d;
6 } myStruct;
7
```

2

STRUCT



```
8 union {
9 char a[12];
10 int b;
11 short int d[4];
12 } myUnion;
13
14 printf("%i", sizeof(myStruct));    14
15 printf("%i", sizeof(myUnion));    12
```

Zeichnen Sie die unterschiedlichen Speicheraufteilung der Variablen innerhalb der vorherigen Beispiele `myStruct` und `myUnion`:

7    Define (1 Punkt)

Erklären Sie wie `define` in C funktioniert. Welchen Wert folgendes Programmstück ausgibt?

```
1 #define callA callB(5)
2
3 void callB(int a) {
4 printf("%i\n", a*2);
5 }
6
7 int main() {
8 callA;
9 return EXIT_SUCCESS;
10 }
```

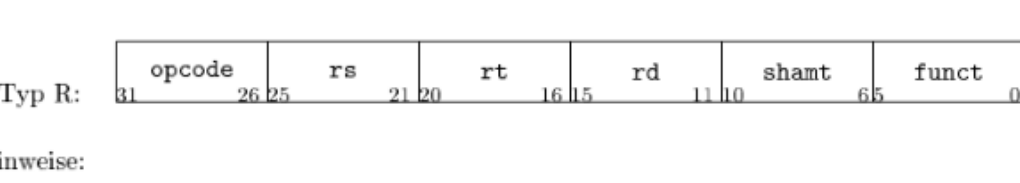
DURCH #define callA callB(5) WIRD ÜBERALL IM CODE  
callA MIT callB(5) ERSETZT => AUSGABE : 10

3

Programmierteil

Die folgenden Aussagen beziehen sich auf die Datei `cSerie1.c`, diese kann von ILIAS herunterladen werden. Ihre Aufgabe ist es, das gegebene Programmgerüst wie folgt zu vervollständigen:

- (a) Laden Sie die Datei `cSerie1.c` von ILIAS herunter und studieren diese aufmerksam. Versuchen Sie zu verstehen, was die bereits vorhandenen Teile bedeuten.
- (b) Tragen Sie zuerst Ihren Namen sowie den Namen einer allfälligen Übungspartnerin oder eines allfälligen Übungspartners an der vorgesehenen Stelle in der Datei ein.
- (c) Erstellen Sie drei Bitfelder namens `InstructionTypeI`, `InstructionTypeJ` und `InstructionTypeR`, die wie folgt aufgebaut sind:



Hinweise:

- [http://publications.gbdirect.co.uk/c\\_book/chapter6/bitfields.html](http://publications.gbdirect.co.uk/c_book/chapter6/bitfields.html)
- Benutzen Sie `typedef`.
- Dasjenige Element, das den niedrigwertigsten Bits entspricht (z.B. `funct` beim Typ `R`), soll an der ersten Stelle stehen.

- (d) Erstellen Sie eine `union` namens `Instruction` mit den folgenden drei Feldern:

- `i` vom Typ `InstructionTypeI`
- `j` vom Typ `InstructionTypeJ`
- `r` vom Typ `InstructionTypeR`

Hinweise:

- [http://publications.gbdirect.co.uk/c\\_book/chapter6/unions.html](http://publications.gbdirect.co.uk/c_book/chapter6/unions.html)
- Benutzen Sie `typedef`.

- (e) Erstellen Sie eine Aufzählung namens `InstructionType` die folgende Elemente umfasst: `iType`, `jType`, `rType`, `specialType`

Hinweise:

- [http://publications.gbdirect.co.uk/c\\_book/chapter6/enums.html](http://publications.gbdirect.co.uk/c_book/chapter6/enums.html)
- Benutzen Sie `typedef`.

- (f) Erstellen Sie eine Struktur namens `Operation`, die folgende Elemente enthält:

- einen String der Länge `OP_NAME_LENGTH` namens `name`
- einen `InstructionType` namens `type`
- einen Zeiger auf eine Funktion namens `operation`, die als einzigen Parameter einen Zeiger auf eine `Instruction` und leeren Rückgabewert hat

Hinweise:

- [http://publications.gbdirect.co.uk/c\\_book/chapter6/structures.html](http://publications.gbdirect.co.uk/c_book/chapter6/structures.html)
- [http://publications.gbdirect.co.uk/c\\_book/chapter5/function\\_pointers.html](http://publications.gbdirect.co.uk/c_book/chapter5/function_pointers.html)
- Benutzen Sie `typedef`.

- (g) Erstellen Sie eine Struktur namens `Function`, die folgende Elemente enthält

3

- einen String der Länge `FUNC_NAME_LENGTH` namens `name`
- einen Zeiger auf eine Funktion namens `function`, die als einzigen Parameter einen Zeiger auf eine `Instruction` und leeren Rückgabewert hat

Hinweise analog vorheriger Teilaufgabe.

- (h) Implementieren Sie die Funktion `printInstruction`. Diese soll eine `Instruction` formatiert ausgeben. Die Formatierung unterscheidet sich je nach Typ der zugehörigen `Operation`:

- iType** • Name der `Operation` als linksausgerichteter String mit Feldbreite 4
  - `rt` und `rs` jeweils als vorzeichenbehaftete Ganzzahlen der Länge 2 mit führenden Nullen aufgefüllt
  - `immediate` als Hexadezimalzahl der Länge 4, ebenfalls mit führenden Nullen aufgefüllt und mit `0x` beginnend
  - Zeilenumbruch
- jType** • Name der `Operation` als linksausgerichteter String mit Feldbreite 4
  - `address` als Hexadezimalzahl der Länge 8, ebenfalls mit führenden Nullen aufgefüllt und mit `0x` beginnend
  - Zeilenumbruch
- rType** • Name der (zugehörigen) *Funktion* als linksausgerichteter String mit Feldbreite 4
  - `rd`, `rs` und `rt` jeweils als vorzeichenbehaftete Ganzzahlen der Länge 2 mit führenden Nullen aufgefüllt
  - `shamt` als Hexadezimalzahl der Länge 4, ebenfalls mit führenden Nullen aufgefüllt und mit `0x` beginnend
  - Zeilenumbruch
- specialType** • Name der `Operation` als linksausgerichteter String mit Feldbreite 4
  - Zeilenumbruch

Hinweise:

- Die der `Instruction` zugehörige `Operation` erhalten Sie mit `Operation o = operations[i->i.opcode]` analog für Funktionen
- [http://publications.gbdirect.co.uk/c\\_book/chapter3/flow\\_control.html#section-5](http://publications.gbdirect.co.uk/c_book/chapter3/flow_control.html#section-5)
- [http://publications.gbdirect.co.uk/c\\_book/chapter9/formatted\\_io.html](http://publications.gbdirect.co.uk/c_book/chapter9/formatted_io.html)
- Als Beispiel liegt der gewünschte Output des fertigen Programmes in der Datei `output.c1` bei. Stellen Sie sicher, dass Ihr Programm diesen Output erzeugt.

- (i) Stellen Sie sicher, dass Ihr Programm mit dem Befehl `gcc -ansi -pedantic -Wall -o cSerie1 cSerie1.c` ohne Fehler und Warnungen kompiliert. Dies ist eine notwendige Voraussetzung, damit der Programmierteil als erfüllt gilt.

- (j) Benennen Sie die Datei `<name1>_<name2>.c` (wobei `<name>*` natürlich durch Ihren Nachnamen zu ersetzen ist).

- (k) Geben Sie die Datei elektronisch durch Hochladen in ILIAS ab.

5