RA FS 22 Series 1

Abdelhak Lemkhenter, Alp Eren Sarı, Sepehr Sameni

The due date for the first series is Tuesday, 15. March 2022 at 3 p.m.. Submit the solution to the theoretical part as a PDF and all source files of the programming part as a zip-file on ILIAS. If questions arise, you can use the ILIAS forum at any time. Possible insolvable tasks are to be disclosed as soon as possible, we will gladly help you.
Have fun!

Theoretical Questions

Total score: 12 points

1 String (1 Point)

Given the code snippet below, what is the content of the memory block associated with variable *word*.

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

2 Accessing arrays using pointers (1 Point)

Given the following `setValueAt` function which updates the value stored in a `double`-array at a given position, provide the equivalent implementation using direct pointer arithmetic. Identify and **fix** potential problems of the provided snippet of code.

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

3 Pointer types (2 Points)

Given the following code:

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

which leads to the following memory content (excerpt):

Address	Content (Hex)
...	
bffff604	00
bffff605	ca
bffff606	9a
bffff607	3b
bffff608	d2
bffff609	02
bffff60a	96
bffff60b	49
...	

Hint: Byte order is Little-Endian, meaning the least significant byte occupies the lowest memory address.

What is the output of the following code and **explain why** (assume a 32-bit architecture, i.e. `sizeof(int) = sizeof(long) = sizeof(void*)`, and that `char` is signed):

```
1  void * p = &b;
2  printf("%x\n", p);
3  printf("%x\n", *(long*)p++);
4  printf("%x\n", *(char*)p++);
5  printf("%x\n", *(unsigned char*)p++);
6  printf("%x\n", p);
```

4 Parameter passing (2 Points)

Given the following program, what are the values of the variables `i` and `j` after program execution?

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

What would be the value of `i` und `j` if the function `increment` was defined as follows?

```
*7 int increment(int *x) {
*8   return (*x)++;
*9 }
```

5 Pointer arithmetic (2 Points)

State the output of each call of `printf` in the code fragment below.

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

State the output of the following program. What possible problems could arise?

```
1  short x = 3;
2  short *px = &x;
3  *(px--) = 20;
4  *px = 21;
5  printf("%i %i\n", x, *px);
```

6 Structs and Unions (3 Points)

Which value will the following code fragment output?

```
1  struct {
2    char a[10];
3    char b;
4    char c;
5    short int d;
6  } myStruct;
7
8  union {
9    char a[12];
10   int b;
11   short int d[4];
12 } myUnion;
13
14 printf("%i", sizeof(myStruct));
15 printf("%i", sizeof(myUnion));
```

Draw the different memory distributions of the variables `myStruct` and `myUnion` from the example above.

7 Define (1 Point)

Explain how `define` works in C. Which value the following program fragment will output?

```
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 }
```

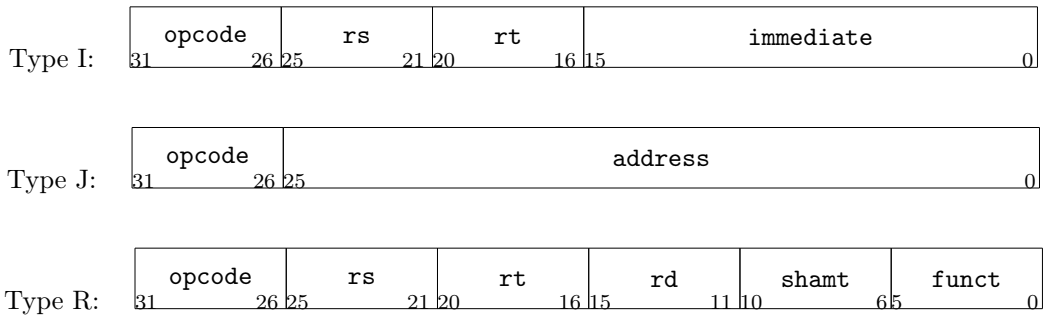
Programming exercise

The following statements refer to the file `cSerie1.c` which can be downloaded from ILIAS. Your task is to complete the given framework as follows:

(a) Download the file `cSerie1.c` from ILIAS and take a thorough look at it. Try to make sense of what is already provided.

(b) Enter your name and the name of your parter in the predefined area of the file.

(c) Create three bit fields named `InstructionTypeI`, `InstructionTypeJ` and `InstructionTypeR`, each composed as follows:



Hints:

- `http://publications.gbdirect.co.uk/c_book/chapter6/bitfields.html`
- Use `typedef`.
- The element corresponding to the least significant bit (i.e. `funct` in Type R) must appear first.

(d) Create a `union` named `Instruction` with the following three fields:

- `i` of type `InstructionTypeI`
- `j` of type `InstructionTypeJ`
- `r` of type `InstructionTypeR`

Hints:

- `http://publications.gbdirect.co.uk/c_book/chapter6/unions.html`
- Use `typedef`.

(e) Create an enumeration named `InstructionType` composed of the following elements: `iType, jType, rType, specialType`

Hints:

- `http://publications.gbdirect.co.uk/c_book/chapter6/enums.html`
- Use `typedef`.

(f) Create a structure named `Operation` composed of the following elements:

- a string of length `OP_NAME_LENGTH` named `name`
- an `InstructionType` named `type`
- a pointer to a function named `operation` with empty return value (void). The only parameter of this function is a pointer to an `Instruction`.

Hints:

- `http://publications.gbdirect.co.uk/c_book/chapter6/structures.html`
- `http://publications.gbdirect.co.uk/c_book/chapter5/function_pointers.html`
- Use `typedef`.

(g) Create a structure named `Function` composed of the following elements:

- a string of length `FUNC_NAME_LENGTH` named `name`
- a pointer to a function named `function` with empty return value. The only parameter of this function is a pointer to an `Instruction`.

Use the same hints as in the previous task.

(h) Implement the function `printInstruction` which outputs an instruction with the correct formatting. This formatting depends on the type of the corresponding operation:

iType
- name of the operation as a left-aligned string, occupying a field of 4 characters
- `rt` and `rs` each as a signed integer of width 2 with leading zeros
- `immediate` as a hexadecimal integer of length 4, also with leading zeros and a `0x` prefix
- line break

jType
- name of the operation as a left-aligned string, occupying a field of 4 characters
- `address` as a hexadecimal integer of length 8, also with leading zeros and a `0x` prefix
- line break

rType
- name of the corresponding *function* as a left-aligned string, occupying a field of 4 characters
- `rd`, `rs` und `rt` each as a signed integer of width 2 with leading zeros
- `shamt` as a hexadecimal integer of length 4, also with leading zeros and a `0x` prefix
- line break

specialType
- name of the operation as a left-aligned string, occupying a field of 4 characters
- line break

Hints:

- You can get a reference to the operation corresponding to the instruction `i` using `Operation o = operations[i->i.opcode]`
  Analogous for functions
- `http://publications.gbdirect.co.uk/c_book/chapter3/flow_control.html#section-5`
- `http://publications.gbdirect.co.uk/c_book/chapter9/formatted_io.html`
- Have a look at the file `outputc1`. The output of your program should look exactly the same.

(i) Make sure that your program compiles without any errors or warnings using the command `gcc -ansi -pedantic -Wall -o cSerie1 cSerie1.c`. This is an essential requirement to pass the programming exercise!

(j) Rename the file to `<lastname1>_<lastname2>.c` (where `<lastname*>` is to replaced with your last name).

(k) Hand in your solution electronically by uploading the file to ILIAS.