RA FS 22 Series 2

Abdelhak Lemkhenter, Alp Eren Sari, Sepehr Sameni

Due date for this second series is Tuesday, 29. March 2022 at 3 p.m.. Please only upload the solution to ILIAS (preferably as a .zip file). If questions arise, you can use piazza at any time. Possible difficult tasks should be disclosed as soon as possible, we will gladly help you.
Have fun!

Theoretical Questions

Total score: 11 Points

1 Function Pointer (1 Point)

What is the output of the following code fragment:

```
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);
```

2 Const (2 Points)

Describe the different properties of the four following declarations.

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

3 Arrays (1 Point)

What is the problem with the following program fragment?

```
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  }
```

4 MIPS Branch Instructions (2 Points)

Assume that register `$s2` contains the value 4, register `$s1` contains the value 1 and register `$s3` contains the value 20.

Using equivalent C code, describe what the following example does:

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

5 MIPS More Branch Instructions (1 Point)

How does the assembler expand the following pseudo-instruction?

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

6 Endianness (1 Point)

Assume that a 32 bit `integer` is stored as a `word` at the address 10001:

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

What is the (decimal) value of the stored number and what is the address of its *least significant byte* if

(a) Big Endian

(b) Little Endian

is used as endianness?

7 Array access (2 Points)

Translate the following C code fragment to Assembly code:

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

Assume that the address of the first element of the array is stored in register `$t2`, `index` in `$t1` and `mul` in `$t4`

8 MIPS Register/System memory (1 Point)

Let B be an array with 11 data words (in system memory) whose base address is stored in `$s1`. Load the last word of B using one single instruction into the register `$s2`.

Optional Questions

The following questions refer to the files from the programming exercise. You do not have to answer them, but they may give you a better understanding of the programming part.

Byte order

What endianness (Byte order) does our MIPS simulation use (Big- or Little-Endian)? Justify your answer.

Declarations

What is being declared here? What are we going to use it for later? (in `mips.h`)

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

Sign Extension

Describe the purpose of the function `word signExtend(halfword value)` (in `mips.c`)

Tests

Describe the tests that are already implemented (i.e. `void test_addi()` in `test.c`)

Programming exercises

Your task is to complete the given framework as follows:

(a) Download the previously mentioned files from ILIAS and take a thorough look at them. Try to make sense of what is already provided. Solving the optional questions above will help you with this.

(b) Enter your name and the name of your partner into the predefined area of the files `mips.c` and `test.c`.

(c) Implement the function `storeWord` (in `mips.c`).

(d) Write sensable and extensive tests for the following MIPS operations (in `test.c`)

`lw`, `ori` and `sub`

(e) Implement the following MIPS operations according to the specifications stated in the Book "Computer Organization and Design" by D.A. Patterson and J.L. Hennessy (in `mips.c`).

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

You may also find the specifications in the PDF "MIPS Reference Data" which you can download from ILIAS (Literatur.zip)

(f) Make sure your implementation will compile without any errors or warnings. You can check this with `make`.

(g) Make sure your implementation passes the given tests without any errors or warnings. You can check this with `make test`

(h) Create a ZIP file from your solution and name it `<lastname>.zip` (where `<lastname>` is to replaced with your last name).

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