Programming 2 - SS23

Project 1 - IBAN-Calculator

Authors: Katharina Buchthal, Christopher Cohnen

26. April 2023

Universität des Saarlandes

Overview

1. Git introduction
2. MARS/MIPS Introduction
3. Tests and Debugging
4. About the project

Git introduction

Configuration

`$ git config` is used to configure Git repositories.

- `--global` sets up the global configuration.
  - `user.name "firstname lastname"`
  - `user.email "...@stud.uni-saarland.de"`

Configuration

`$ git config` is used to configure Git repositories.

- `--global` sets up the global configuration.
  - `user.name "firstname lastname"`
  - `user.email "...@stud.uni-saarland.de"`

Example

```
$ git config --global user.name "Konrad Klug"
```

Git project repository

We can obtain the project using `$ git clone` and the following url:

```
ssh://git@git.prog2.de:2222/project1/<NAME>.git
```

<NAME> = your username on the Prog2-CMS

Git project repository

We can obtain the project using `$ git clone` and the following url:

```
ssh://git@git.prog2.de:2222/project1/<NAME>.git
```

<NAME> = your username on the Prog2-CMS

Caution

You must have created and uploaded an ssh-key to git.prog2.de beforehand.

▸ How to activate the GitLab account and add a newly generated ssh key

Submitting the project

- `$ git status` - list modified files
- `$ git add <file>` - stage the modified files
- `$ git commit -m "message"` - commit all the staged files
- `$ git push` - submit commits

Submitting the project

- `$ git status` - list modified files
- `$ git add <file>` - stage the modified files
- `$ git commit -m "message"` - commit all the staged files
- `$ git push` - submit commits

Caution

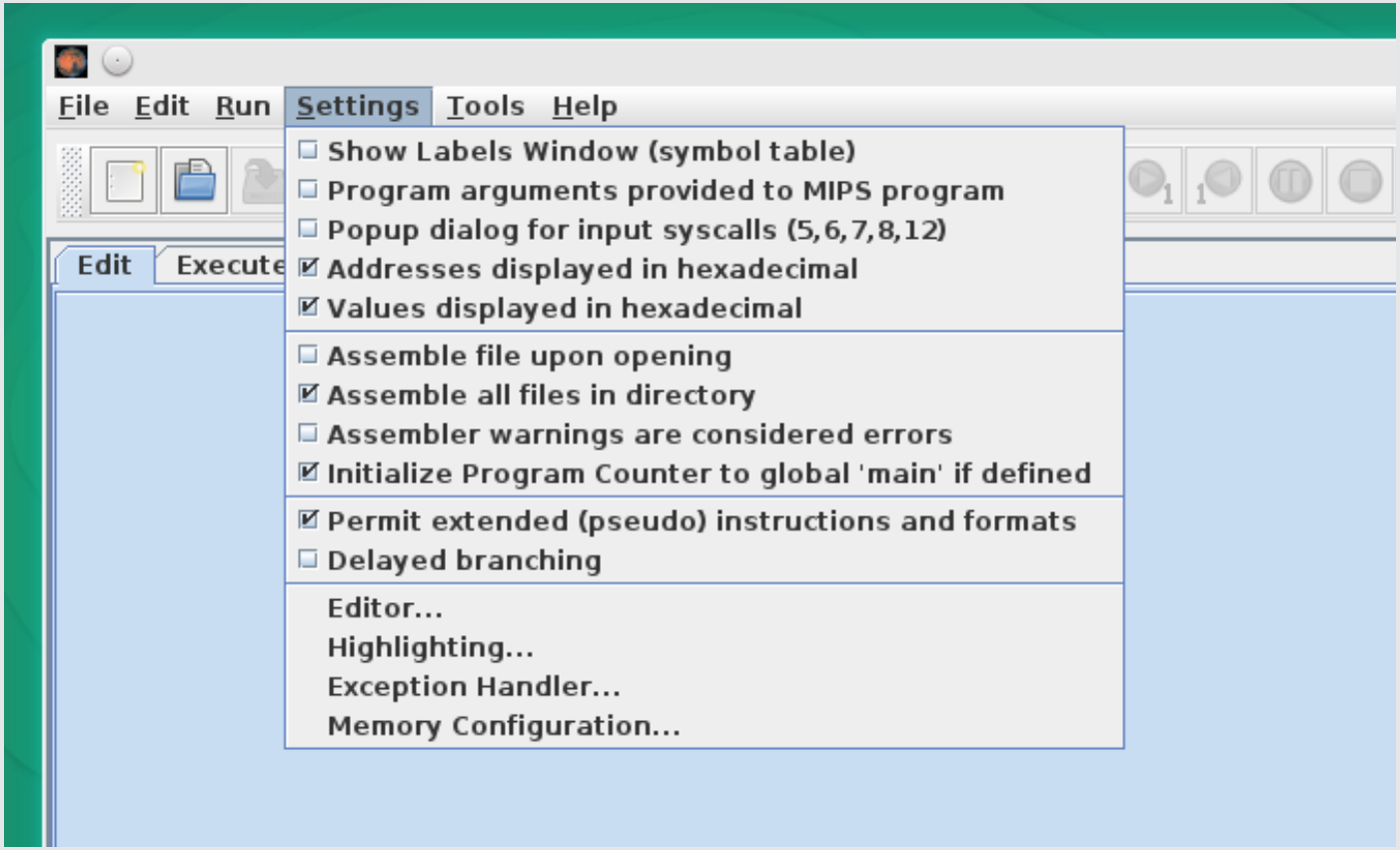
Only the changes you submitted by Tuesday, 9th May 2023, 23:59 onto the server are tested and counted as valid submissions.

MARS/MIPS Introduction

MARS Settings

Caution

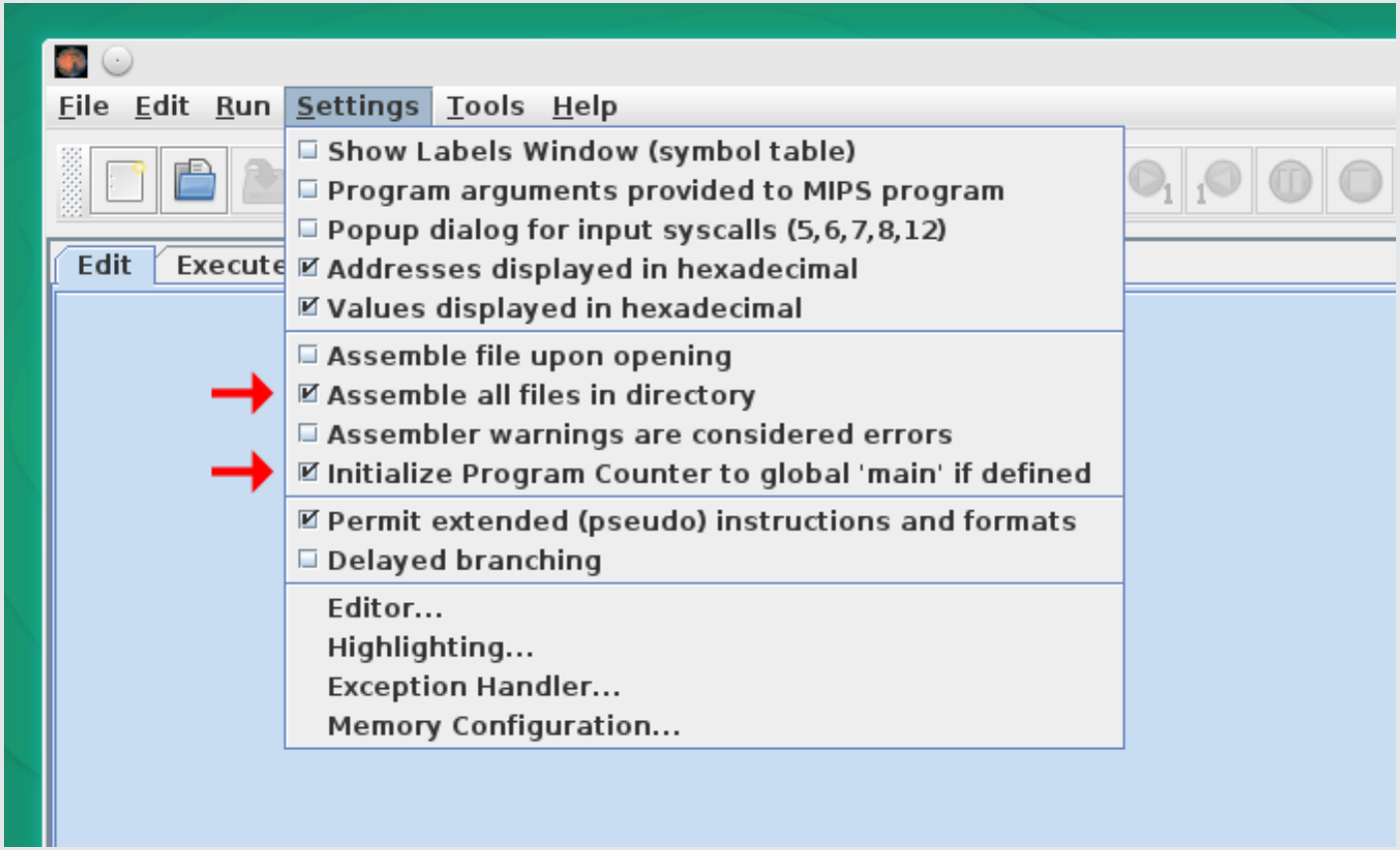
We have to adjust two settings:



MARS Settings

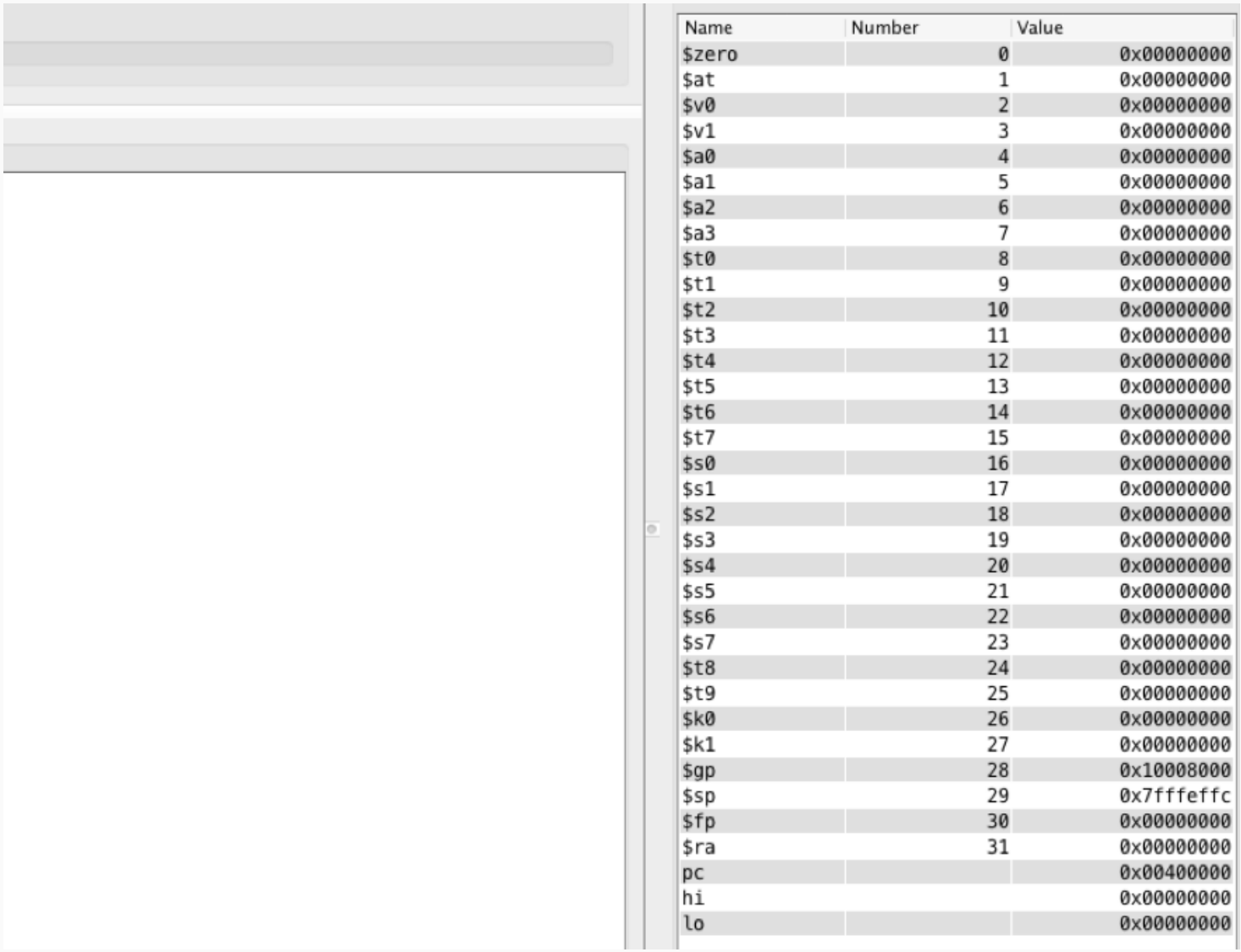
Caution

We have to adjust two settings:



Calling Conventions

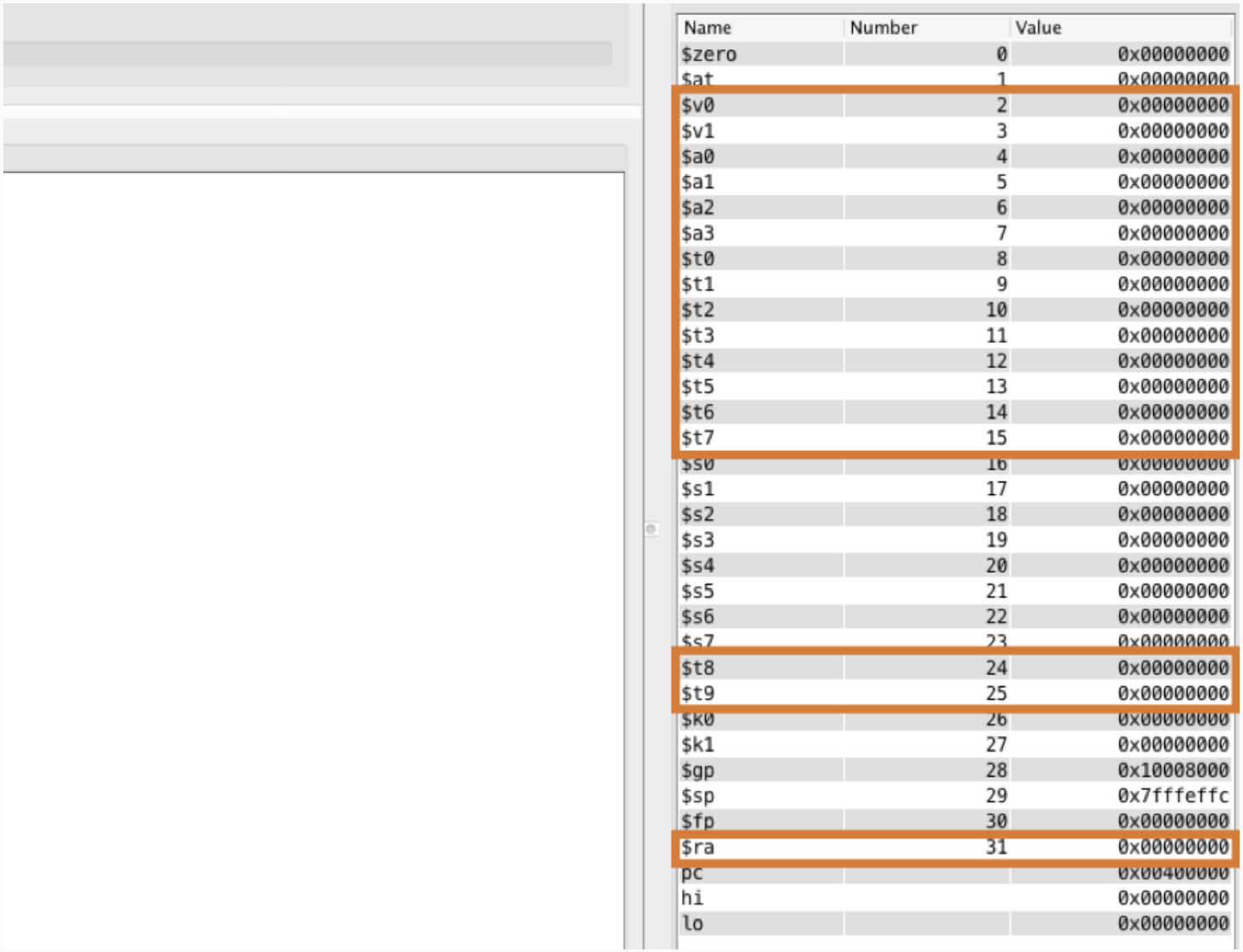
Registers



Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

Calling Conventions

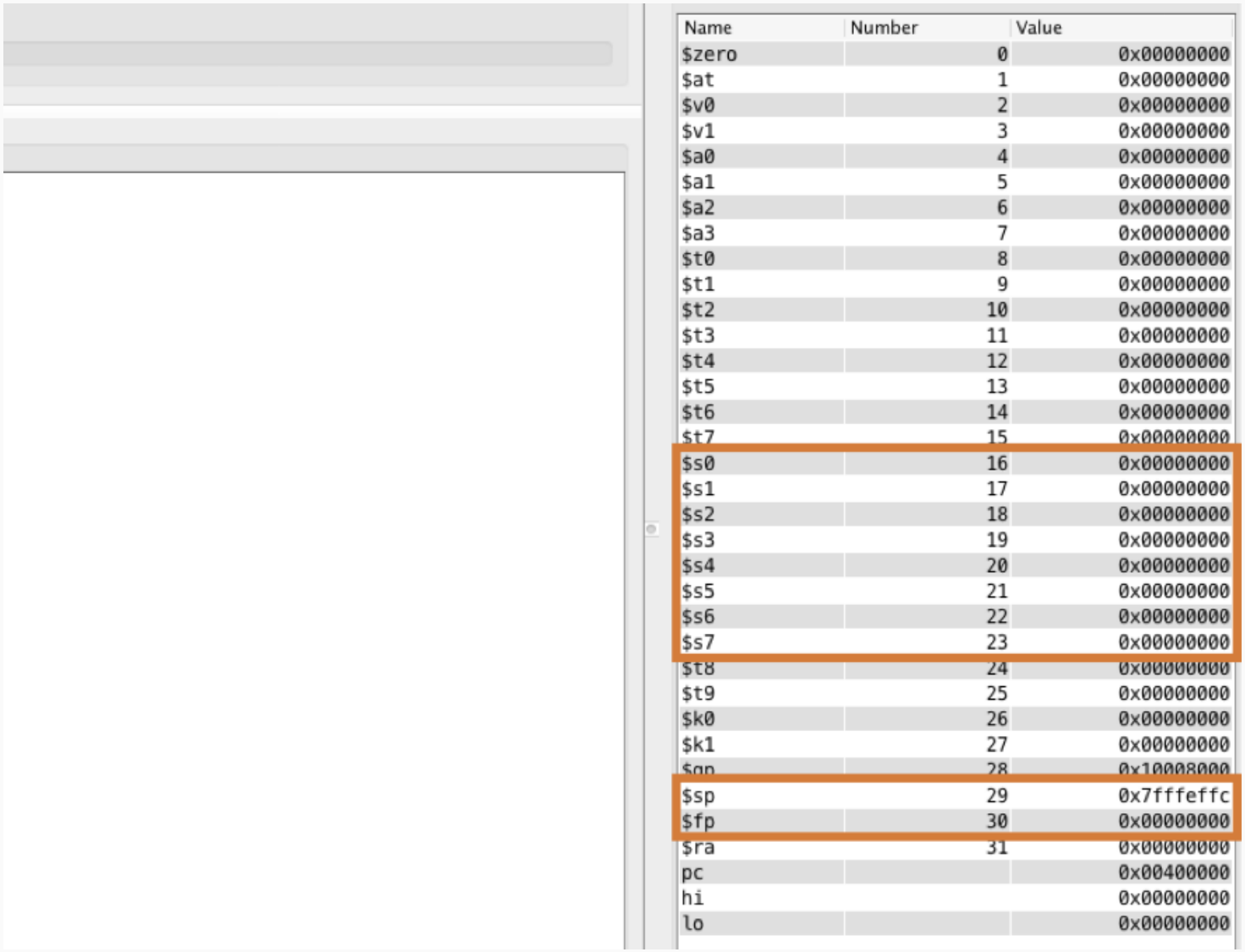
Registers - Caller save



Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

Calling Conventions

Registers - Callee save



Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

Tests and Debugging

Tests

- Public Tests - come with the project, can run locally
- Daily Tests - run at least once a day after you have pushed something onto the server
- Eval Tests - run after submission

Tests

- Public Tests - come with the project, can run locally
- Daily Tests - run at least once a day after you have pushed something onto the server
- Eval Tests - run after submission

Caution

All public tests for a subtask must be passed in order to receive points for that subtask.

Public Tests

We can run the Public Tests in our project folder using `$ ./run_tests.py`.

Most important arguments

- `-h` : list of all possible arguments
- `-l` : list of all tests
- `-t test1` : execute only `test1`
- `-d dir` : execute tests in directory `dir`
- `-b` : execute the tests with your bonus task implementation

Writing own tests

We can create our own tests in a `tests/student`-folder:

- Create *asm*-file containing the test (must have a global `main` label)
- Create *ref*-file containing the expected output
- Run `$ ./run_tests.py -d student`

Debug tests

We can debug Public Tests using
`$ ./build_testbox tests/pub/testfile.asm`.

We can then execute and debug the files in the `testbox` directory.

Bonus Tasks

`$ ./build_bonusbox tests/pub/testfile.asm`
works in the same way but copies the file
`bonus/validateChecksum.asm` into the bonusbox.

Example - Modulo calculation

We want to calculate $$a0 \mod 11$$ and return the result within $v0.

Code

```
        .globl modulo_str
        .text
    modulo_str:
        li      $t1   11
        divu    $a0   $t1
        mfhi    $v0

        jr      $ra
```

Example - Testing

test.asm

```
        .globl main
        .text
    main:
        li      $a0   42
        jal     modulo_str
        move    $a0   $v0
        li      $v0   1
        syscall
        li      $v0   10
        syscall
```

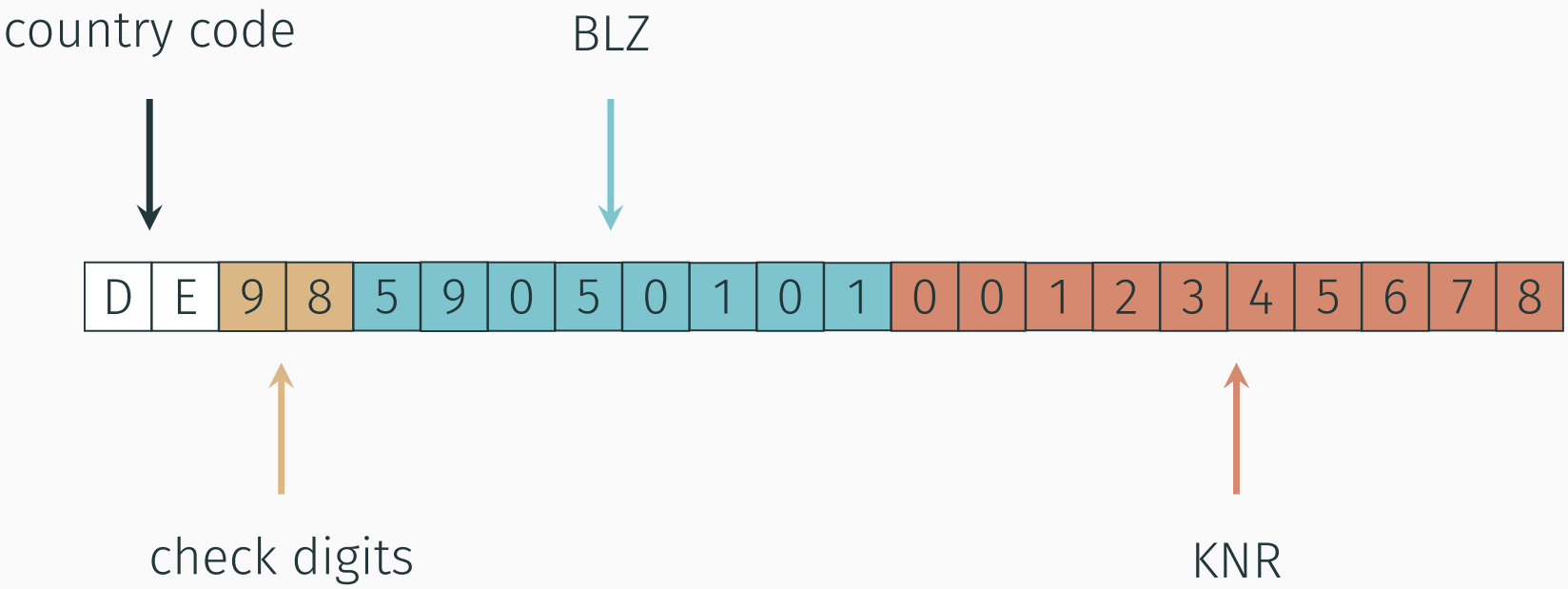
test.ref

```
    9
```

Questions?

About the project

IBAN


country code
BLZ
D E 9 8 5 9 0 5 0 1 0 1 0 0 1 2 3 4 5 6 7 8
check digits
KNR

Task 1

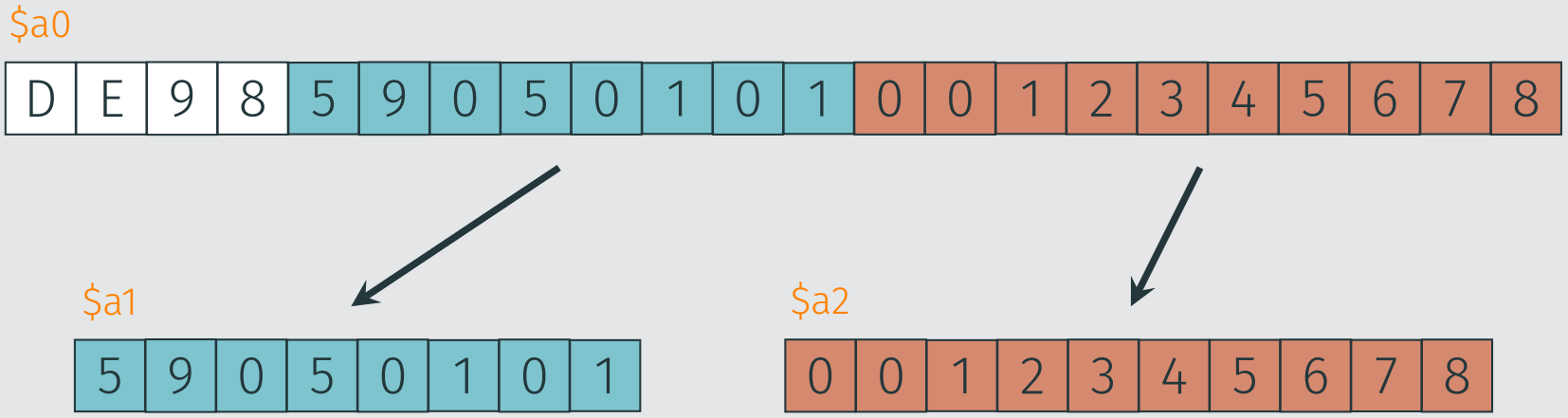
In the file `iban2knr.asm`:

Read KNR and BLZ from IBAN

- $a0 - address of IBAN buffer (22 bytes) **(read)**
- $a1 - address of BLZ buffer (8 bytes) **(to fill)**
- $a2 - KNR buffer address (10 bytes) **(to fill)**

Task 1 - Example

Functionality



Caution

The digits in the buffer are encoded as ASCII characters.

Task 2

In the file `moduloStr.asm`:

Calculate remainder

- $a0 - start address of buffer
- $a1 - number of characters in buffer
- $a2 - the divisor
- $v0 - return value (calculated remainder)

Caution

The digits in are the buffer encoded as ASCII characters.

Task 2 - Example

Register content

- $a0 : `0xC0FEBABE`
- $a1 : `4`
- $a2 : `17`
- $v0 : `10`

Functionality

Problem

Complete IBAN is too large for 32-bit registers.

Horner's method

Solution: Going step by step

$$1 \cdot 1000 + 2 \cdot 100 + 3 \cdot 10 + 4 = 1234$$

$$((1 \cdot 10 + 2) \cdot 10 + 3) \cdot 10 + 4 = 1234$$

Horner's method

Solution: Going step by step

$$1 \cdot 1000 + 2 \cdot 100 + 3 \cdot 10 + 4 = 1234$$

$$((1 \cdot 10 + 2) \cdot 10 + 3) \cdot 10 + 4 = 1234$$

Modulo rules

$$(a \cdot b) \mod c = ((a \mod c) \cdot b) \mod c$$

$$(a + b) \mod c = ((a \mod c) + b) \mod c$$

Horner's method

Solution: Going step by step

$$1 \cdot 1000 + 2 \cdot 100 + 3 \cdot 10 + 4 = 1234$$
$$((1 \cdot 10 + 2) \cdot 10 + 3) \cdot 10 + 4 = 1234$$

Modulo rules

$$(a \cdot b) \mod c = ((a \mod c) \cdot b) \mod c$$
$$(a + b) \mod c = ((a \mod c) + b) \mod c$$

Hint

You find a code example in chapter 3.4 of the Prog2 book.

Task 3

In the file `validateChecksum.asm`:

Checksum for correctness

- $a0 - Address of a string representing a German IBAN (22 characters)
- $v0 - Return value (calculated remainder)

Task 3 - Example

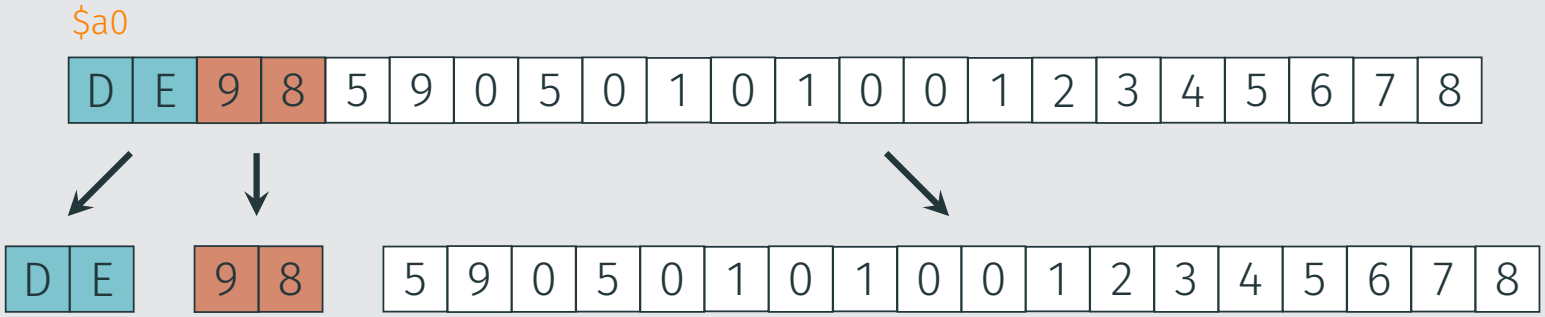
Functionality

Caution

There might not be enough space behind the IBAN buffer to just copy the first digits to the back.

Task 3 - Example

Functionality

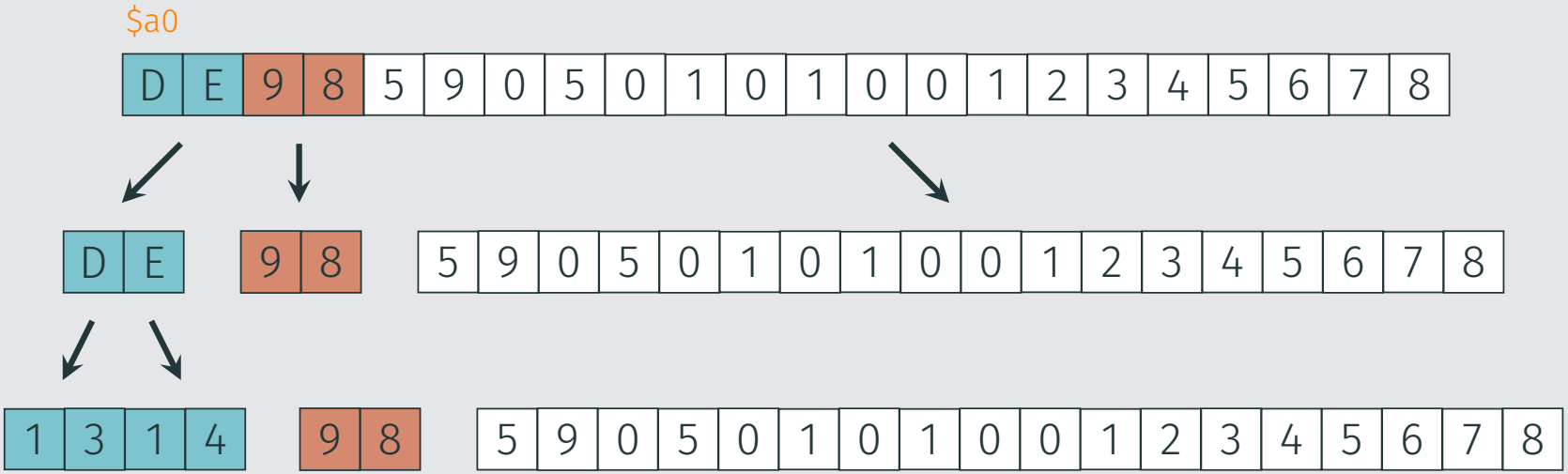


Caution

There might not be enough space behind the IBAN buffer to just copy the first digits to the back.

Task 3 - Example

Functionality

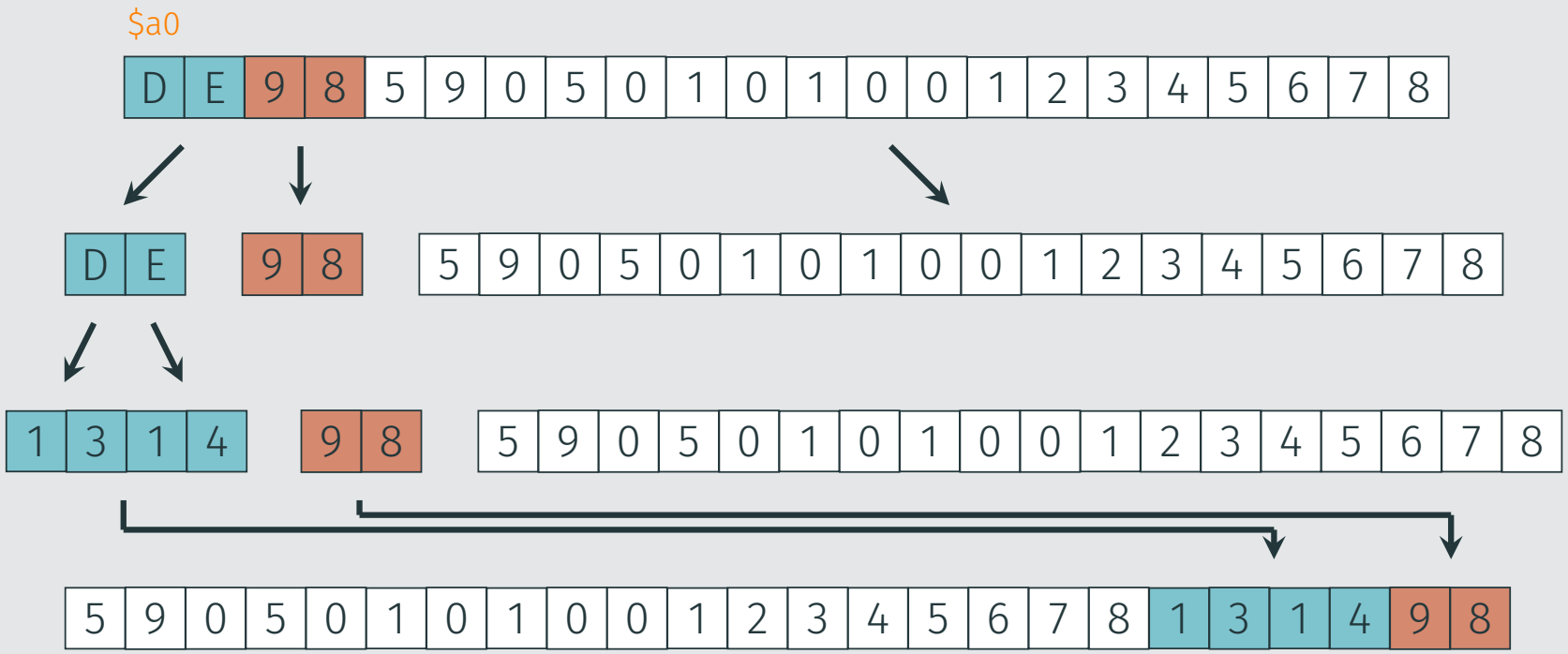


Caution

There might not be enough space behind the IBAN buffer to just copy the first digits to the back.

Task 3 - Example

Functionality

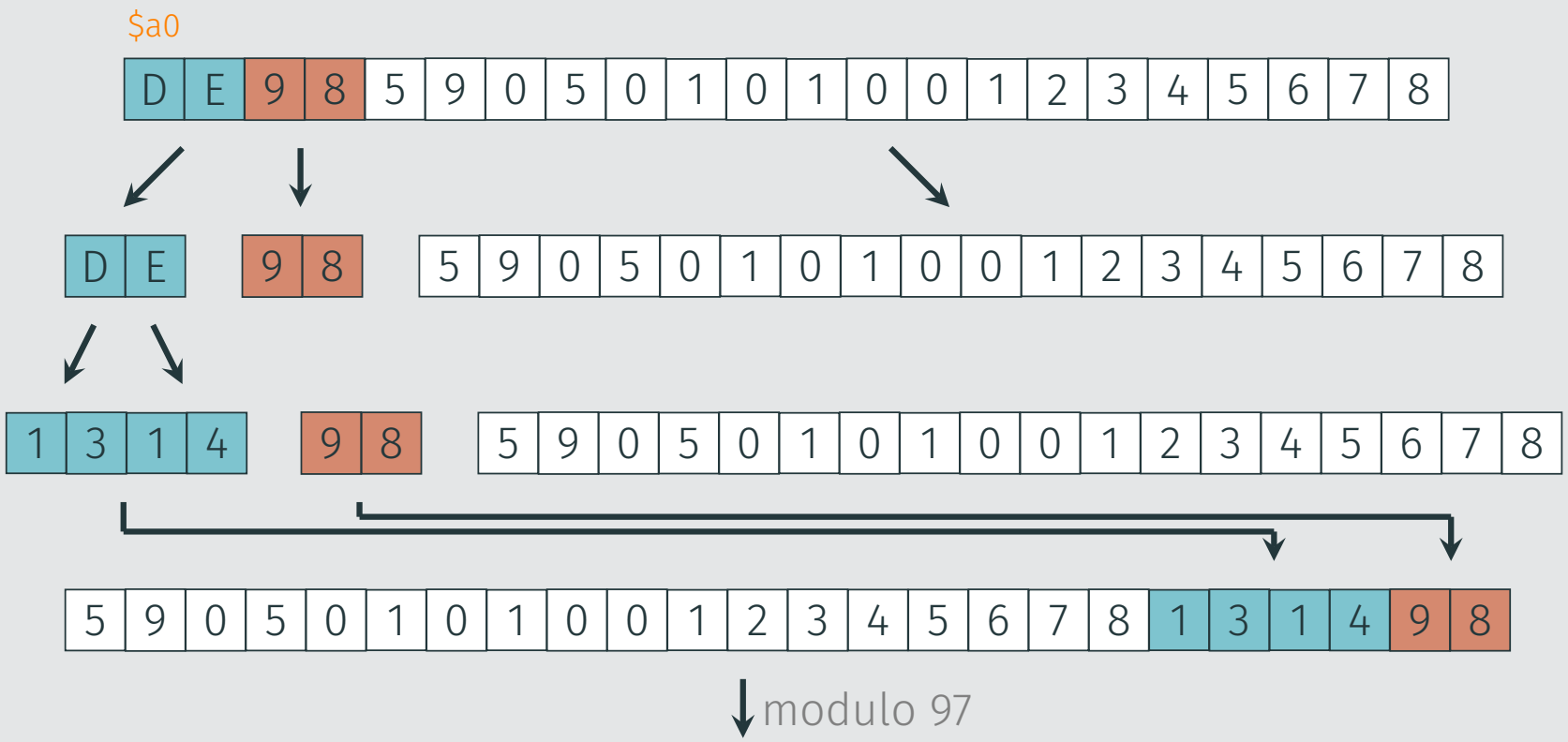


Caution

There might not be enough space behind the IBAN buffer to just copy the first digits to the back.

Task 3 - Example

Functionality



Caution

There might not be enough space behind the IBAN buffer to just copy the first digits to the back.

Task 4

In the file `knr2iban.asm`:

calculateIBAN

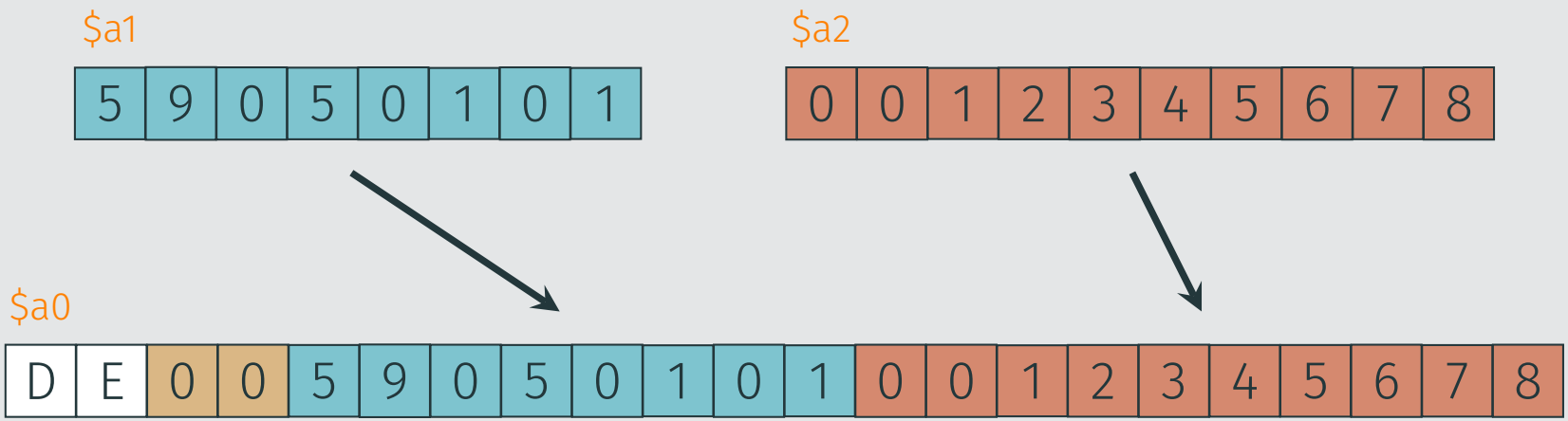
- $a0 - start address of IBAN buffer (22 characters)
- $a1 - start address of BLZ buffer (8 characters)
- $a2 - start address of KNR buffer (10 characters)

Task 4 - Example

Functionality

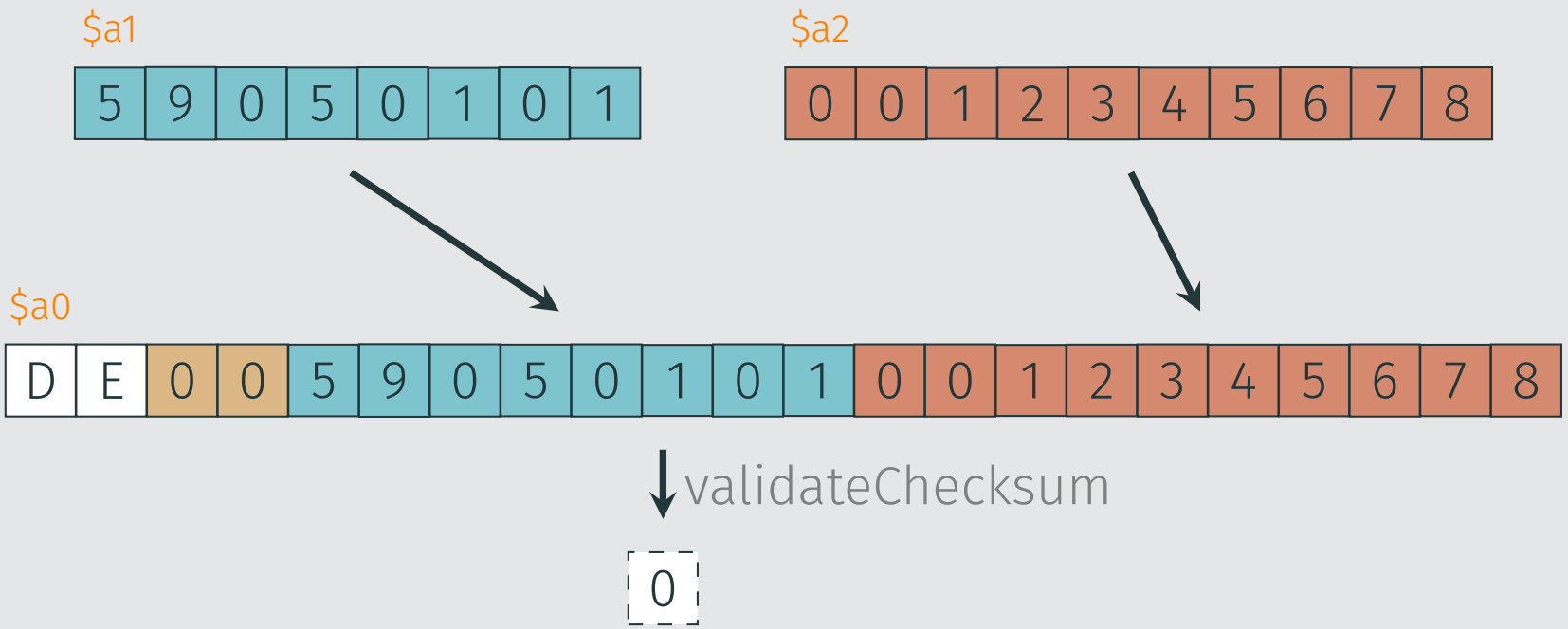
Task 4 - Example

Functionality



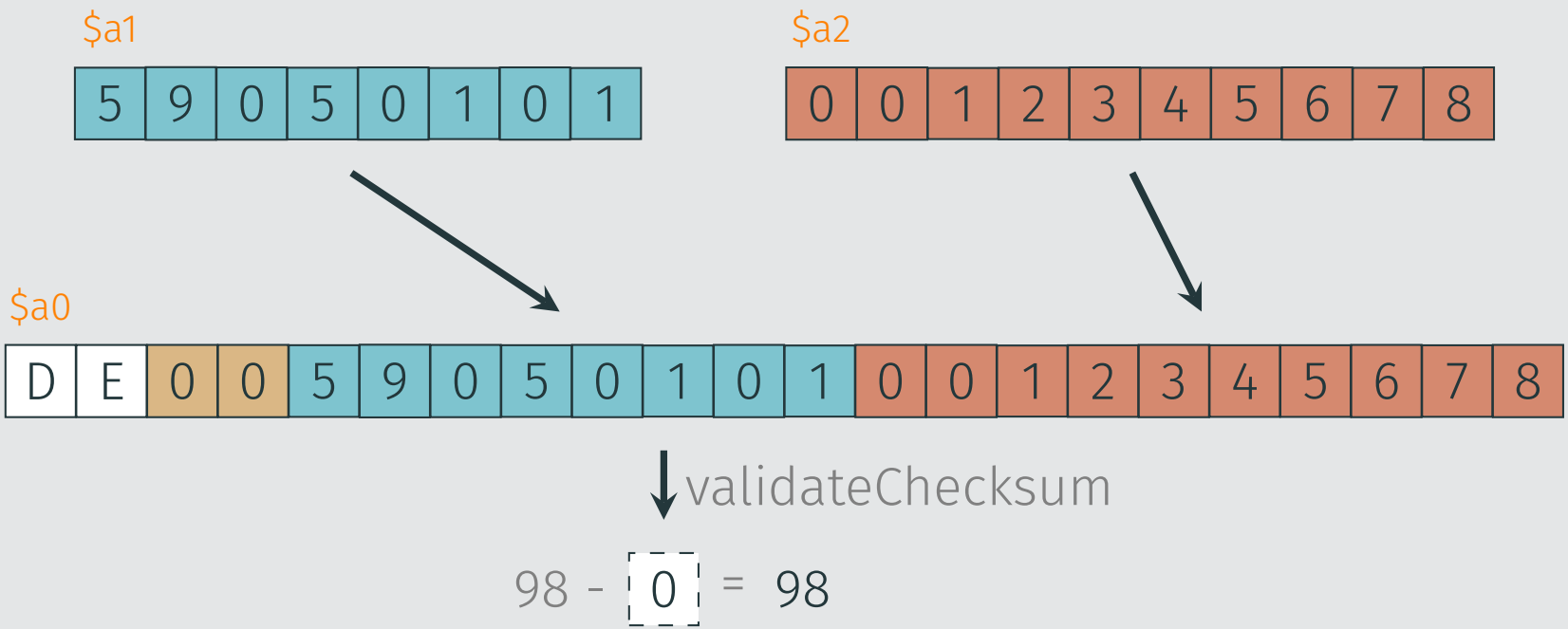
Task 4 - Example

Functionality



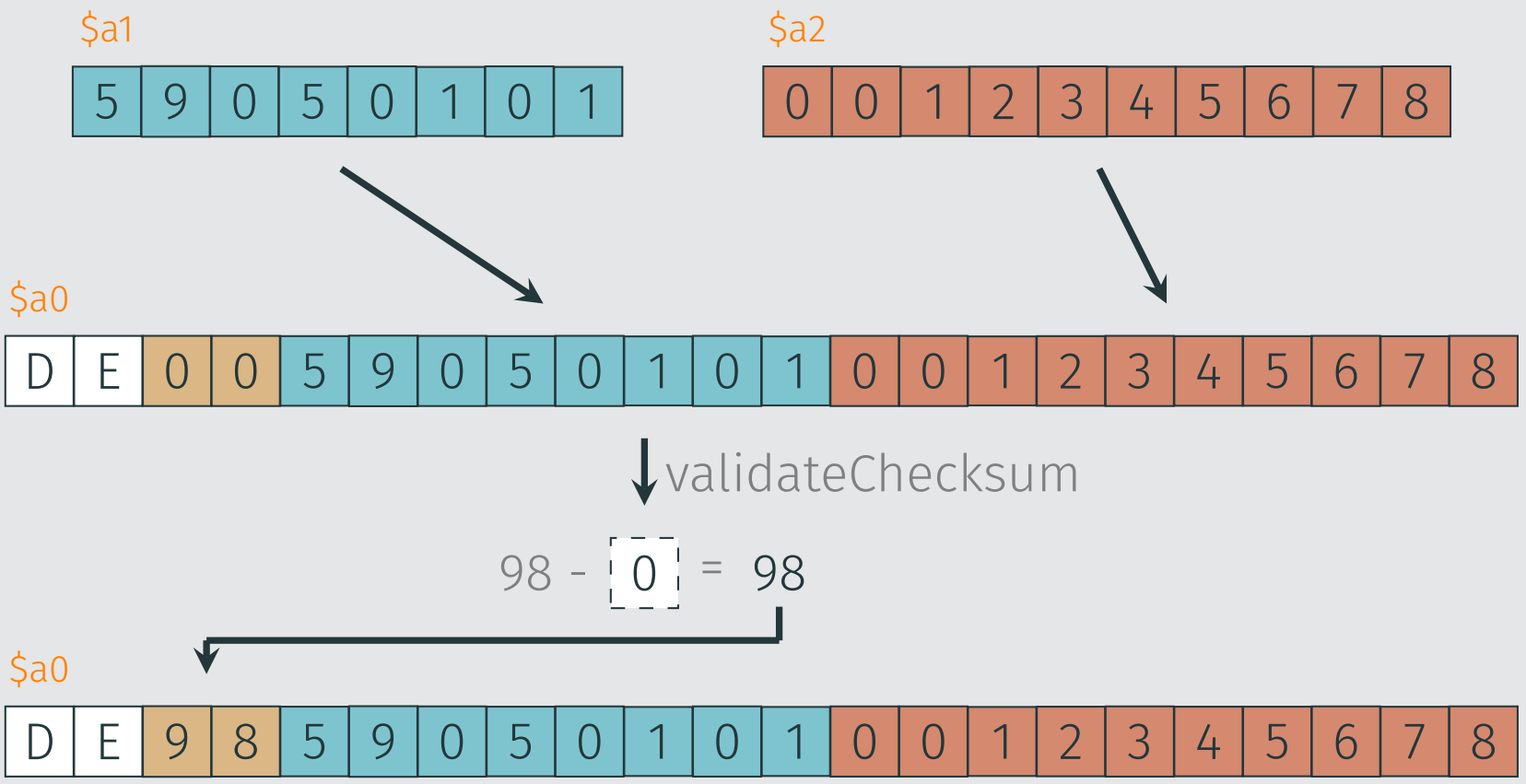
Task 4 - Example

Functionality



Task 4 - Example

Functionality



Bonus Task

In the file `bonus/validateChecksum.asm`:

Validate checksum

Same as the normal validateChecksum task, BUT:

- without writing to memory
- without using syscalls

Caution

When calling a function, you normally need to store something to fulfil the calling convention. This is also not allowed here!

Helper functions

In the file `util.asm` there are pre-implemented helper routines.

For example, to copy memory areas and to save a number as an ASCII character.

Please use them!

Questions?

If you have any problems, use the forum or come to the Office Hours!