Laboratory Practice

Design of a 4 bit CPU

Writing Program Code

During the practice the task is to create own program code running on the CPU. The instruction memory of the CPU (INST_MEM) is described initially by a VHDL code (INST_MEM.vhd). It is possible to overwrite the content of given cells manually with the hexadecimal code of the instructions. This way it is possible to edit the instructions in the memory of processor.

The easiest way to create program code is to define them with their mnemonics. The code written this way should be processed by a program, which compiles it to binary code. The program code will be created during laboratory practice with the program **dt4bit_CPU_assembler**[1] (Fig. 1.). This program is capable to edit the program code, to compile it to binary code and to replace the default content of the program memory. After the content of program memory is changed (INST_MEM.vhd), then the project has to be translated again with the ISE Webpack. This way there is possibility to simulate the operation and to upload the configuration file to the FPGA. The FPGA programmed this way will contain in its program memory the modified program code.

On the editorial interface can the program code be edited, this is where the instructions have to be defined. The accepted instructions can be fined in the section **List of accepted instructions** (Elérhető utasítások listája). On the editorial interface only the following characters may be used:

- lower case letters of English alphabet (a..z),
- numbers (0..9),
- colon, to assign label (:),
- tabulator to separate instruction (TAB),
- enter to separate line (ENTER).

Lines should be separated the following way:

label: ⟶ instruction ⟶ argument

(címke: ⟶ utasítás ⟶ argumentum)

[1] The program is at the moment available only in Hungarian. Therefore, the Hungarian texts are listed everywhere with their translations. Also a brief dictionary can be found at the end of the document, where translation of the most important phrases can be found.

It is optional to assign label and argument. If the instruction doesn't have argument, then a new line have to be introduced after the instruction. If the line doesn't contain a label, then line have to begin with tabulator. Otherwise, the label has to be assigned with a colon at the beginning of the line. The length of the label may be 3 characters maximum, without the colon. The different parts (label, instruction, argument) has to be separated with tabulators.

It is possible to **save** (Mentés) or **open** (Megnyitás) an existing program at the **Source File operations**.

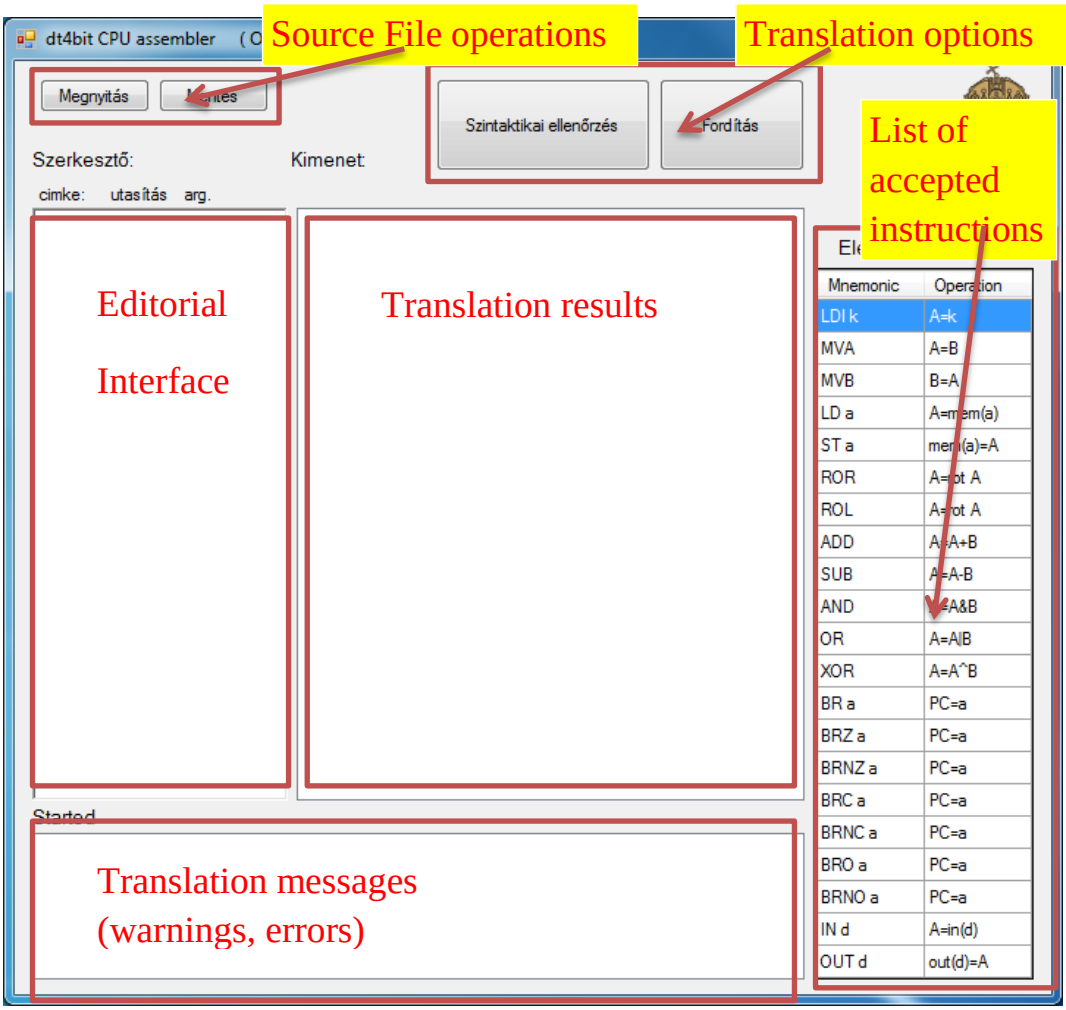

Figure 1. Structure of the program code editor

The **Translation options** contains two buttons. The first is the syntax check (Szintaktikai ellenőrzés), which check the syntax of program code in the Editorial Interface. The possible information messages appear in the **Translation messages** window. If there is a syntax error, the wrong line could be highlighted by clicking on the error message. The instruction, its hexadecimal code and its location in the program memory is displayed in the **Translation results** window.

By selecting the Translation (Fordítás) button, at first a syntax check will be done. IF the program code is syntax error free, than the translator replaces the content of the instruction

memory (INST_MEM.vhd) with the actual source code. In this case the message *"Syntax check and Translation finished!"* appears. *("Szintaktikai ellenőrzés és Fordítás kész!")*

The first task is to run the **assemblerDTcpu4.exe** file in the folder of the project. Open the **prog/example_for.txt** file. This example realize a for loop. The first instruction read the input value of input0, than set the out0 display to zero value. Finally, the value of out0 will increase until it reaches the value of input0. Translate the program with the ISE Webpack, create the binary code for programming the FPGA, and upload this code to the development board. Analyse the operation of the program.

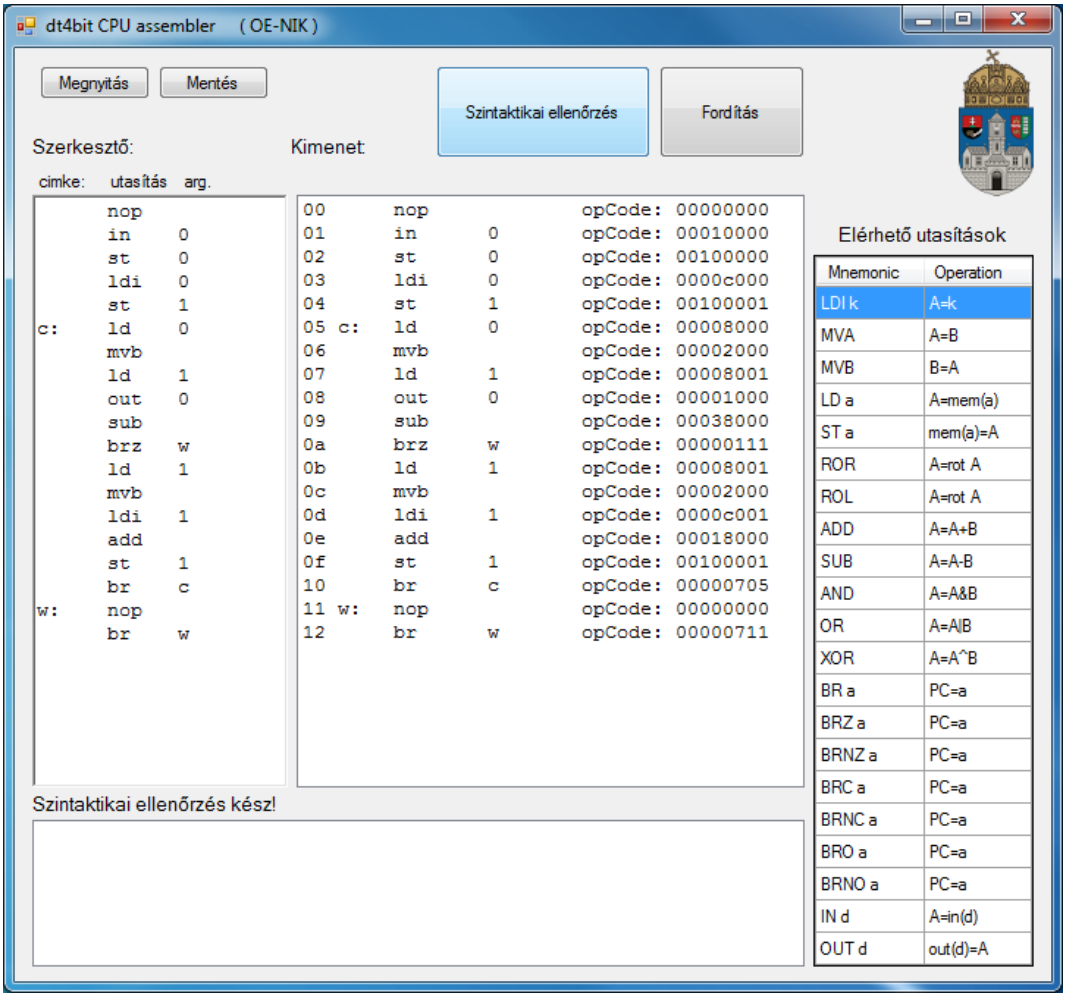

Figure 2. Example of a for loop

The C language equivalent of source code:

```
1  unsigned int v;
2  v = in(0);
3  for(unsigned int i = 0, i <= v, i++)
4  {
5      out(0) = i;
6  }
7  while(true) {}
```

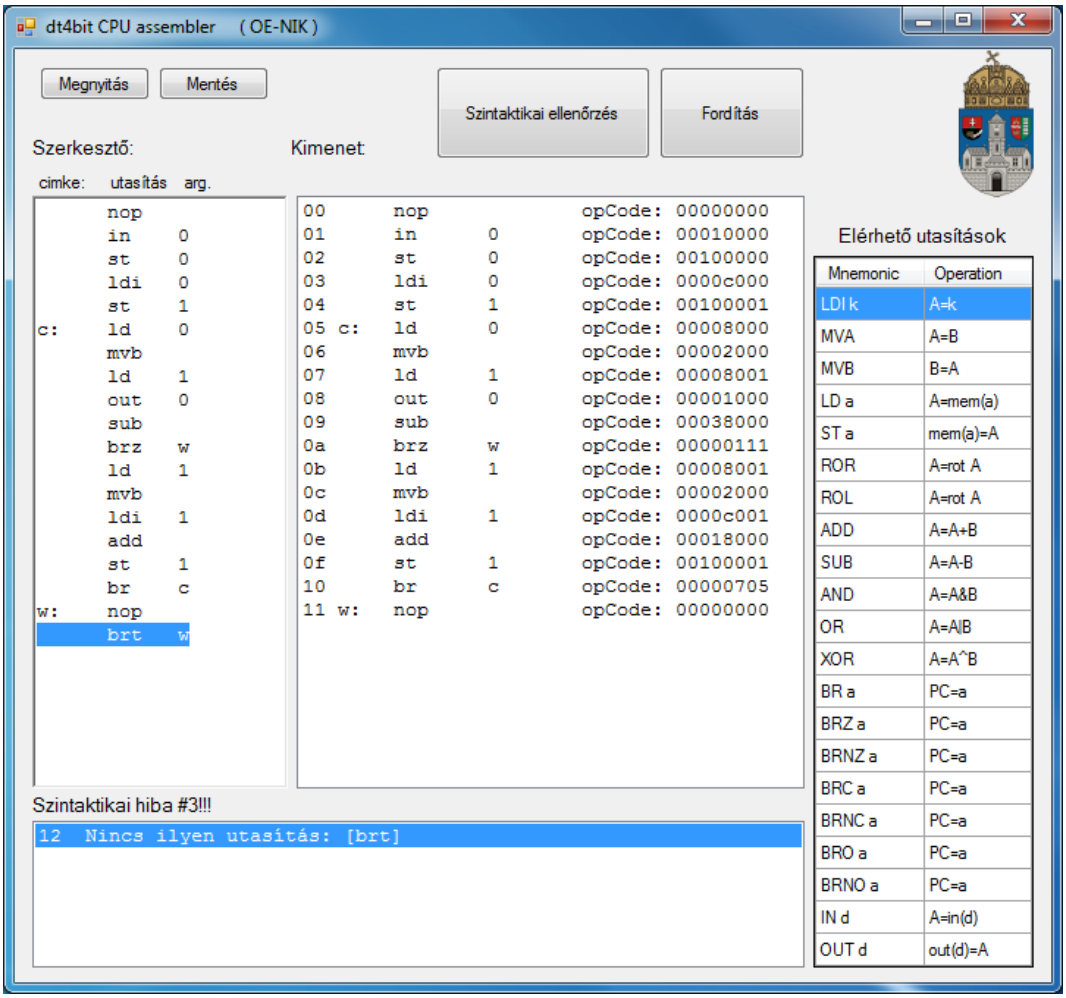

Figure 3. Example of for loop with error message

Dictionary

Megnyitás - Open

Mentés - Save

Szintaktikai ellenőrzés – Syntax check

Fordítás - Translation

Elérhető utasítások – Accepted instructions

Szerkesztő - Editor

címke - label

utasítás - instruction

arg. - argument

Kimenet - Output

Szintaktikai hiba […] – Syntax error […]

Szintaktikai ellenőrzés kész! - Syntax check and Translation finished!

Nincs ilyen utasítás […] – Instruction […] not found