Homework Assignment – Computer Architecture

2023/2024

For the homework assignment in the Computer Architecture course, Peter Confusington needs to implement a compiler for the assembly language of the simple 16-bit Mini MiMo computer using the ARM assembly language. The model of the Mini MiMo computer includes 4 16-bit registers (R0, R1, R2, and R3). Instructions and memory words are also 16-bit. Arithmetic and logical instructions are 2-operand. The addresses of the Mini MiMo computer start at 0x0000.

As Peter is not proficient in programming, he asks for your help. Write a program that, given a program in ASCII string source_code, calculates the symbol table for the Mini MiMo computer program. Each new line in the string is marked with the ASCII character '\n' or LF. The directive ".var" defines a 16-bit unsigned or signed number. Each integer or command can only be labeled with one label. Spaces or tabs are allowed before the label. The label must end with the ':' character. You can assume that the program is syntactically correct.

Implement the program in multiple steps.

In the **first** step, copy the source_code string to the cleaned_source_code string, removing comments and unnecessary spaces. Single-line comments start with symbol '@'.

In the **second** step, remove unnecessary empty lines and rewrite the cleaned_source_code string back to source_code.

For example, consider a program in the assembly language for Mini MiMo. Program lines are numbered sequentially.

```
1.)
2.)
3.)         stev1: .var 0xf123          @ comment 1
4.) @empty line
5.)        stev2: .var     15
6.) stev3: .var 128
7.)_start:
8.) mov r1, #5 @move 5 to r1
9.)mov r2, #1
10.)ukaz3: add r1, #1
11.)b _start
```

After the first and second steps, the source_code string should contain the cleaned source code.

```
1.)stev1: .var 0xf123
2.)stev2: .var 15
3.)stev3: .var 128
4.)_start: mov r1, #5
5.)mov r2, #1
6.)ukaz3: add r1, #1
7.)b _start
```

In the final, **third** step, iterate over the source_code list again and calculate the symbol table, which is implemented as a list. Each label is represented by an ASCIZ string followed by a 16-bit address. If the address in the symbol table is not aligned, write an additional byte with a value of 0. You can check if the value of a register is even or odd using the "tst" instruction.

```
tst r1, #1
bne odd @branch if r1 contains odd number
```

After completing the program, the symbol table should contain the following content. Pay attention to the additional zeros due to terminated strings and aligned addresses.

```
'stev1' 00 00 00 00 'stev2' 00 01 00 'stev3' 00 02 00 '_start' 00 00 03 00 'ukaz3' 00 05 00
```

It is not necessary to complete the entire homework assignment, you can solve individual subproblems. You have to do an oral defence of your solution. Assignment consists of two parts. First is mandatory and with the second (non-mandatory part) you can gain some extra points to your LAB result.

The mandatory assignment will be considered complete if you **complete one of the following options**:

- write a program for the **first** and **second** part, where you clean up unnecessary spaces, tabs, comments, and empty lines from the source code;
- write a program for the **third** part only, assuming that the source code is already cleaned up, and calculate the symbol table;

The non-mandatory assignment will be considered complete if you **complete one of the following options**:

- complete the entire mandatory assignment (**first**, **second**, and **third** part);
- write any other more complex program in assembler based on your own idea;
- write and execute a program based on your own idea on MiniMiMo CPU model or contribute in any other way (modify model, create your own CPU, write assembler, etc.)

```
.text
.org 0x20
source_code: .asciz "           \n\n        stev1: .var 0xf123            @ comment 1\n @empty line \n          stev2: .var      15
\nstev3: .var 128\n_start:\n mov r1, #5 @move 5 to r1\nmov r2, #1\nukaz3: add r1, #1\nb _start"
cleaned_source_code: .space 120
symbol_table: .space 100
.align
.global _start
_start:
            @Write your program here!
_end: b _end
```