# Educational Pascal Compiler into MMIX Code

Evgeny A. Eremin
Perm State Pedagogical University
614990, Russia, Perm,
Sibirskaya, 24
eremin@pspu.ac.ru

## ABSTRACT

The pedagogical software tool, visualizing translation from Pascal into machine codes, is described. The basic aim of the present work was to create software, distinctly demonstrating students the close unity of the contents of two fundamental CS courses – programming and computer architecture.

## Keywords

Education, learning, Pascal, compiler, MMIX.

Intensive development of software tools increases the distance between a real executable program in processor instructions and its text on a high-level language. Specialists with long computer experience have constantly perfected their knowledge parallel to the development of calculating machinery. As they naturally progressed from processor codes to modern languages, they tried the full pallet of programming technologies. But now most people, who first begin studying high-level languages, can't imagine how programs they write are connected with processor instructions described in fundamental CS books.

The professional compilers have no purpose to generate demonstrative code and visualize it before a student. So special educational software [1] developed by the author is offered for these aims. It shows the accordance between any simple Pascal program and its executable equivalent in MMIX code (MMIX is a modern model of RISC computer, designed by Knuth [2,3]). A similar version for Intel processors' code also exists.

The suggested educational freeware for Windows media [1] can translate some subset of Pascal language which includes all basic algorithmical structures and run the result of compilation (virtual MMIX machine is realized inside the software to execute the output code). The compiler processes all standard Pascal data types (integer, real, char, boolean) and arrays of them.

The listed above possibilities allow to explain students the following high-level language features:

- variables, constants and type constants realization and the difference between them;
- memory organization in the form of array;
- converting relational Pascal types into each other;
- algoritmical structures on high and low level

and some other fundamental programming language basics.

Let us consider how to use the compiler for learning array indexing as an example. Figure 1 shows translation of the small demo program that works with arrays.

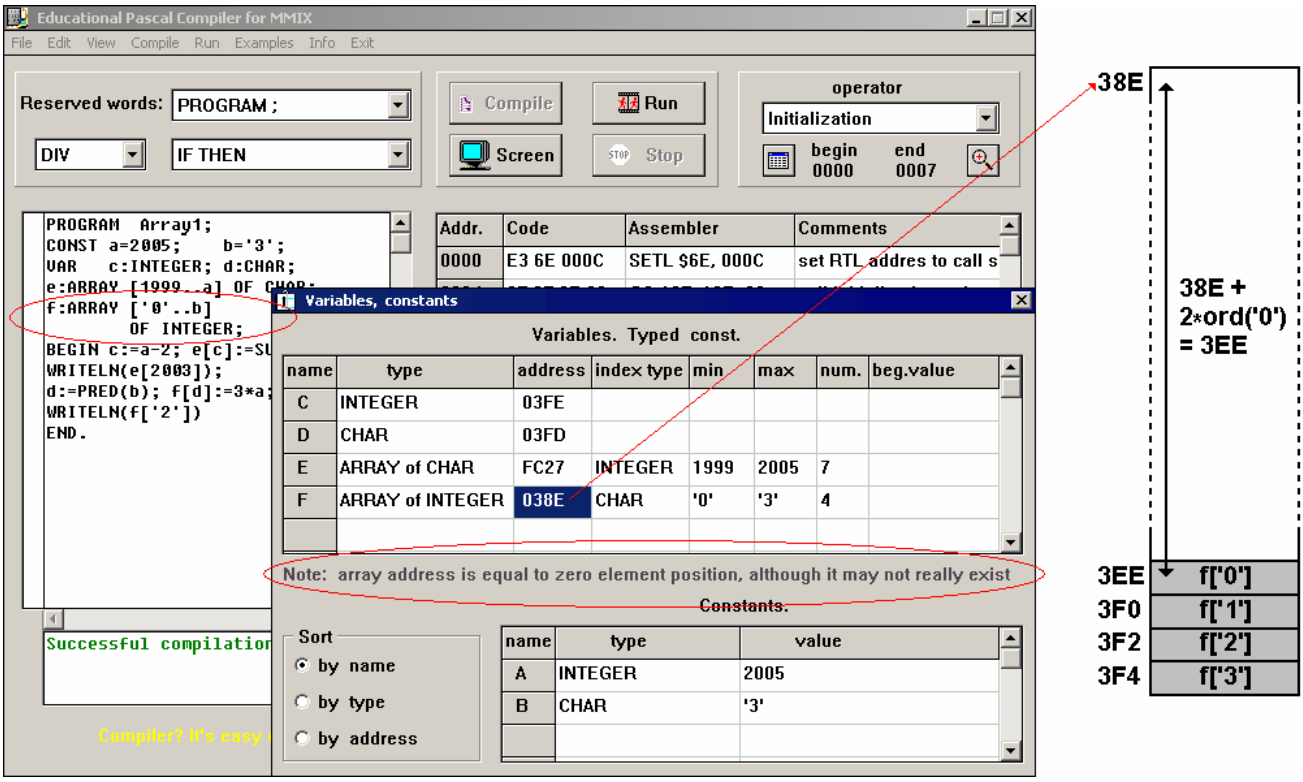


**Figure 1. Using compiler to study arrays.**

Note that this is only one of more than fifty samples built into the software; besides students can type their own programs.

The variables' description in this Pascal program announces that integer array named *f* has one index of char type, and its values belongs to '0'..'3' range. It is much more usable for further calculations to store the address of zero index although such element actually does not exist (in our case address is equal to 38E as it results from Figure 1). Then to calculate current address, for example *f[d]*, compiler can execute the following computation (see Table 1).

**Table 1. Calculations of address for array**

| Address | Code | Assembler | Operation | Comment |
|---|---|---|---|---|
| 104 | E30303FD | SETL $3, 3FD | 3FD → $3 | *d* address |
| 108 | 81030300 | LDB $3, $3, 0 | ($3) → $3 | load value |
| 10C | 39030301 | SL $3, $3, 1 | left shift $3 | *2 - integer |
| 110 | E300038E | SETL $0, 38E | 38E → $0 | 0-base |
| 114 | 20000003 | ADD $0, $0, $3 | $0+$3→ $0 | *f[d]* address |

Analyzing this code, students can clearly see how array indexing is organized in the computer's memory.

The suggested educational compiler can also optionally generate standard fragment of program code that implements reasonability check of index *d* current value. This feature may come in useful for learning compiler fundamentals too.

To demonstrate how simple for understanding the compiled machine code of total Pascal program is, let us review the easiest example of the program here.

```
PROGRAM sample;
CONST x = 2;
VAR   y: INTEGER;
BEGIN y := x*x;
      WRITELN(y)
END.
```

The results of sample's compilation into MMIX code are presented in Table 2 (gray color in the table groups lines with isolated Pascal operators).

**Table 2. Equivalent MMIX program with comments**

| Address | Code | Assembler | Operation | Comment |
|---|---|---|---|---|
| 0 | … | | | RTL library |
| B4 | E30003FE | SETL $0, 3FE | 3FE → $0 | ***y*** address |
| B8 | E3010002 | SETL $1, 2 | 2 → $1 | constant ***x*** |
| BC | 19010102 | MUL $1, $1, 2 | $1*2→$1 | ***x*** * ***x*** |
| C0 | A5010000 | STW $1, $0, 0 | $1 → ($0) | store to ***y*** |
| C4 | E30103FE | SETL $01, 3FE | 3FE → $1 | ***y*** address |
| C8 | 85710100 | LDW $71, $1, 0 | ($1)→$71 | load value |
| CC | 9F6F7010 | GO $6F, $70, 10 | print integer | WRITE ***y*** |
| D0 | 9F6F7034 | GO $6F, $70, 34 | next line | LN |
| D4 | 00000000 | TRAP 0 | exit | END |

The contents of the table seem to be compact and clear. The conventional indirect scheme is used to exchange with any Pascal variable: first put its address into a processor register (marked by '$' in MMIX notation) and then make a reference through it.

We must additionally mention that a little Run-Time Library (RTL) is placed in the beginning of the code. The discussed sample needs RTL for printing the result of calculations to virtual MMIX display. Output RTL subroutines use registers $6F - $71 by definite simple arrangements. To make RTL code shorter and clearer, the realization of some standard algorithms (like converting float or integer numbers into string) are displaced into virtual ROM, so small pieces of code in the library just organize correct subroutine callings.

The software not only shows MMIX code in the above table, but gives some additional possibilities to students: they can extract any operator for examining (including search by run-time error address), analyze the map of variables, inspect system RTL and ROM.

A clear Windows interface allows even the beginners to work with this program. Dropdown list boxes with ready-to-use reserved words and initial standard template for the programs which a student can change make input process quick and comfortable.

One of the software versions has been used for several years at our University. In spite of some non-essential difficulties in the beginning, the results are positive: all students understand the basic compilation ideas. Most of them singly come to the conclusion that universal automatically generated code is not optimal and human translation is better. We discuss this problem and organize the competition for the shortest code with an advanced group of students. Another interesting form of task is to predict the results of compilation without computer and then check these predictions using the compiler.

The traditions of teaching in our country require a profound understanding of learning material. So the described software is meant to extend conventional high-level programming courses first of all. Working with educational compiler, students can acquire a wider outlook in computer languages foundations. Such pedagogical approach closely unites two fundamental CS disciplines – programming and computer architecture, which modern students often misapprehend as unrelated.

## REFERENCES

[1] Educational Pascal Compiler into MMIX Code. http://www.winsite.com/bin/Info?27000000037259.

[2] Knuth, D.E. *A RISC Computer for the Third Millennium.* Heidelberg, Springer-Verlag, 1999.

[3] MMIX 2009 a RISC computer for the third millennium. http://www-cs-faculty.stanford.edu/~knuth/mmix.html.