International Journal on Islamic Applications in Computer Science And Technology, Vol. 4, Issue 1, March 2016, 25-34

Arabic Computer Programming Education Tool

Mohamed Tahar Ben Othman
Computer Science Dept., College of Computer,
Qassim University, KSA
maathaman@qu.edu.sa; mtothman@gmail.com

ABSTRACT

In this paper, we present a programming tool, which is built to help students write their programs in Arabic in a smart environment and execute them directly on the machine or displaying their execution through a simulation. The simulator is used to help students understand the machine's architecture and how programs are internally executed. Over the simulator a small kernel is added to manage a set of programs concurrently executed. This kernel gives an idea to the students of how the operating system is scheduling different programs at different types of priorities. This tool is a part of a project that aims to have an entire environment in Arabic used for teaching several programming languages related courses.

***Keywords*: Arabic programming language, compiler, smart editor, simulator.**

1. INTRODUCTION

Taking courses in one's native language is necessary to develop a sound understanding of sciences. At Arab Universities, computer sciences were formerly taught in Arabic, but the tools' operative instructions were generally presented in foreign languages. The aim of this project is to fill the gap between courses and tools. This paper conveys the general idea behind the project; it describes what has been done and hints to what is remaining. As will be shown in this paper, the goal of this project is not only to build a tool that supports an Arabic programming language, but also an environment which will be used to guide the students through writing programs at the highest possible level until the execution on the machine or an attached simulator. The latter helps the students understand the machine's architecture while executing a program as well as the role of the operating system while scheduling programs. To the best of our knowledge, and besides the fact that the tool is dedicated to Arabic speaking students, there is no tool that guides the students from writing programs to execution while helping them to understand what is given in different programming related courses. The main works found use an interpreter (Galperina, 2013; Al-A'ali, Mansoor and Hamid, 1995; Elsheikh, 2014). (Amin, 2001; Essam, 1996) are implementing an object oriented Arabic language. A Master Thesis (Al-Ethawie, 1997) is a design of a logic Arabic language.

The rest of the paper is organized as follows: the DHAD project's architecture is presented in Section 2. The compiler is explained in Section 3. Section 4 presents the editor and its smart

contribution. The different program execution types are provided in Section 5. The conclusion and future works are discussed in Section 6.

2. ARCHITECTURE OF DHAD PROJECT

As shown in Figure 1 the DHAD project is being built in different phases and consists of several components. The compiler is the heart of the tool. It translates the DHAD programs to machine, C, and Assembly languages. The Editor is smart and helps users to write their programs. The learning component is an interactive user interface that guides the user in learning the DHAD programming through different complexity levels. The Exams component can be used by the Instructor to build exams for his students.

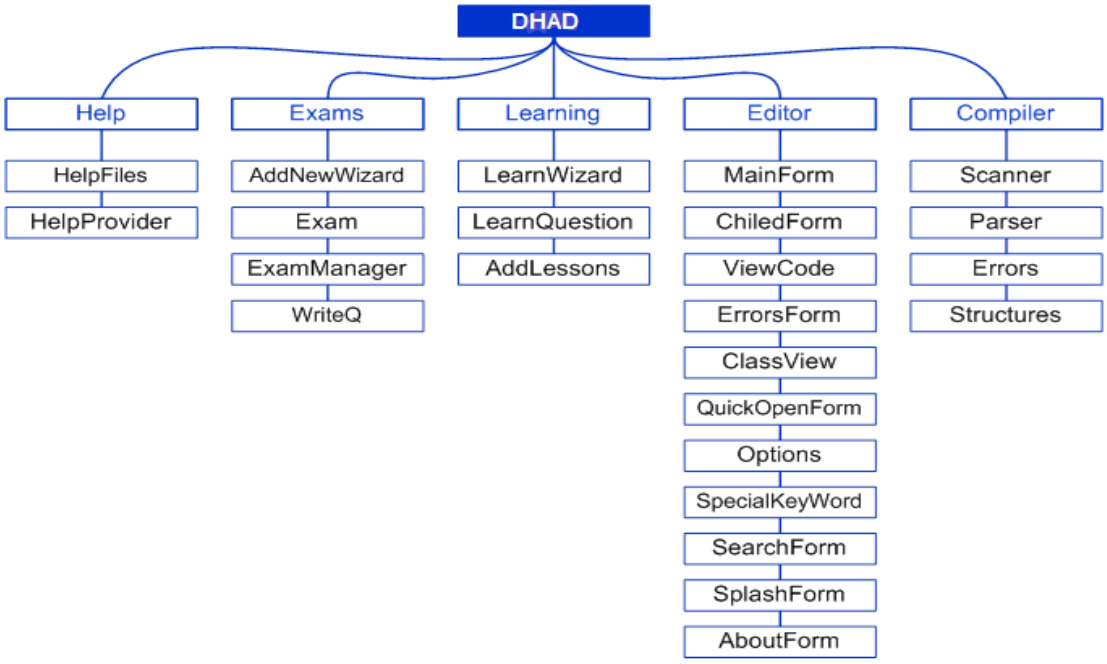

Fig 1: DHAD tool's architecture

3. COMPILER

Prior to build the compiler, the DHAD grammar should be determined. This grammar is chosen as a combination among several high level programming languages' grammars (C, C++ and Pascal). The easiest (the highest level) syntax for instructions and declarations is used for DHAD programming language's grammar. This is the first step toward a user-oriented product.

Figure 2 depicts the compiler's structure. The DHAD programming language compiler has a large number of functionalities that make it an efficient tool for programming studies. The input of the compiler has to be a file with the extension "apl" — Arabic Programming Language — and depending on the compiler options the output can be the filename.c containing the translation of the program written in Arabic into C and/or filename.asm containing the translation of the program into assembly language.

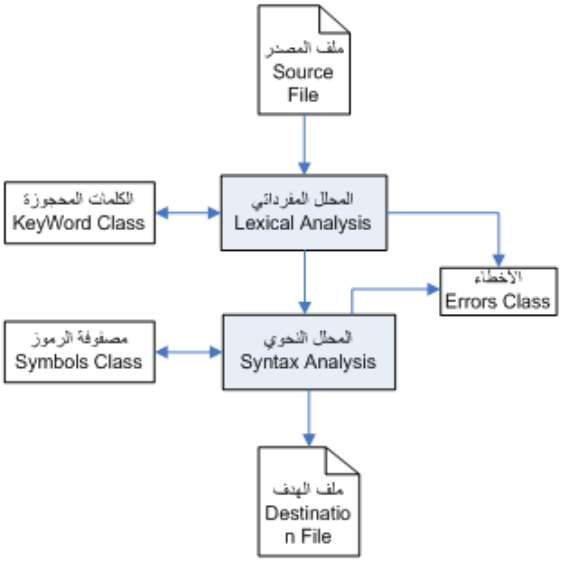

Fig 2: Compiler's structure

3.1 DHAD Language Syntax

A- Keywords:

Table 1 gives some default keywords that are chosen for the DHAD programming language:

Table1 : DHAD Language Keywords

اختبر	اخرى	ادخال	أدخل	ادخلح	ادخلحر
ادخلحرف	إذا	ارجع	اس	اطبع	أعد
الى	أو	بداية	ثابت	جا	جتا
جذر	حالة	حتى	حرف	حفظ	حقيقي
دالة	رئيسي	سجل	سلسلة	صحيح	طالما
طباعة	طول	ظا	ظتا	عالج	عرض
عرضحرف	عرضسلسلة	فارغ	قارن	قف	لا
لصق	لو	مادام	مصفوفة	مطلق	ملف
من	نسخ	نفذ	نهاية	والا	

In the first phase the syntax is taken from three different programming languages (C, C++ and Pascal) and modified, to be easily used and understood later. The following are some examples of instruction syntaxes:

B- For-loop instruction syntax

من <متغير> = <تعبير> **حتى** <الشرط> **خطوة** <قيمة> **عالج** <الأمر>؛

The statement <الأمر> can be either a simple or compound statement. To assist the use of this instruction at a higher level before even studying the variable we also provide another variant of this instruction where there is no need to use the index variable.

من <قيمة> **حتى** < قيمة> **خطوة** <قيمة> **عالج** <أمر>؛

The compiler adds a variable for this for-loop instruction. All precautions are taken to ensure that this variable is not and will never be used by the user and cannot be reused by the compiler in case of nested loops.

C- While-loop instruction syntax

Even if the user can always change keyword lexemes, in some cases, the same instruction is provided using different symbols. Among these cases, the while-loop instruction can be used in two ways:

طالما <الشرط> **عالج** <الأمر>؛

مادام <الشرط> **عالج** <الأمر>؛

D- Input-Output instruction syntax

أدخل صحيح <متغير>؛

أعرض "رسالة"، <نوع> <متغير>؛

This is the short format of the input and output instructions. They can be used with more strings and variables. As the library is not built yet, the DHAD programming language user does not have to include header files at this level. The compiler adds all needed header files. This is another way to increase the language level.

3.2 The Compiler Modules

The compiler consists of several modules described in Figure 2. The lexical analyser is the interface between the user program and the rest of the compiler components. This module's main task is to read a stream of characters from the input program file and to construct a stream of words. A token is assigned to each word and is sent to the next module: the parser. The parser is built based on the grammar of the language. It is the element that verifies if the stream of tokens returned by the lexical analyser respects the grammar rules. By combining the grammar rules and the translation rules, the parser calls upon the third main module which translates to the target language if there is no syntax error in the source program.

4. THE EDITOR

Figure 3 shows the interface of the editor. In addition to standard functionalities the editor introduces new ones to be more user-friendly. Among these functionalities are:

1. Help menu: describes all DHAD programming language instructions' syntaxes with examples that can be copied, compiled and executed.
2. File menu: contains the standard file management functions.
3. Compile menu: this menu contains several options: translation to C, translation to Assembly language and translation to machine language and execute. The interaction between the editor and the compiler is discussed later in this paper.
4. Keywords customization: in Arabic there is no standard technical keywords yet; even if there are some works done in this direction, we included this functionality to give the user the ability to change any keyword in the DHAD programming language. At any time, the user can return back to the default keywords used in the DHAD language. This is another functionality that allows more flexibility in using the tool. Figure 6 gives way to change an existing keyword

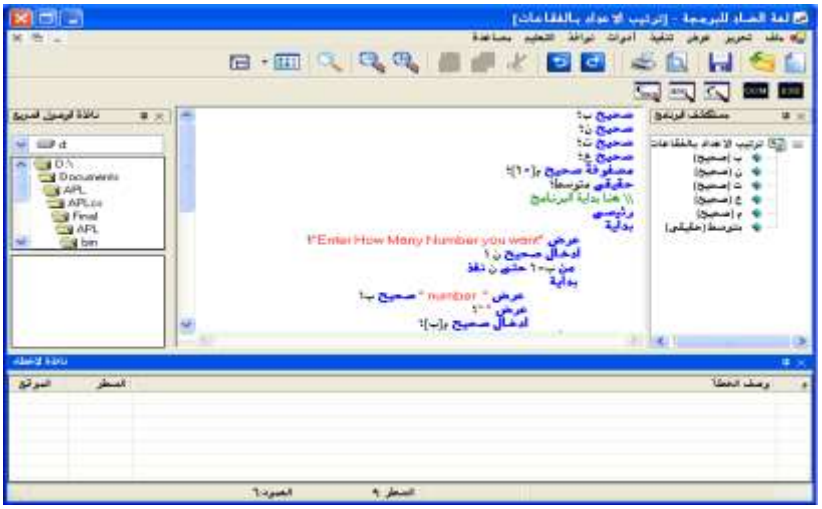

Fig 3: Editor interface

Among the functionalities that are added to the smart editor is the possibility to provide user with a list of the identifiers (names of variables and functions) used in the current program. This will give the user quick access to these identifiers throughout the program. Figure 4 is a screenshot of the identifiers' window. The user can also use his preferred colours for the different parts of the program. Figure 5 shows the different colours used.

Figure 4: Identifiers quick access

Figure 5: Colors specification window

Figure 6: Keywords customization

The program can also be given a name and then inserted in the program each time it is duplicated with small modifications. This will not oblige the user to retype or look for a code to copy and paste.

5. PROGRAM EXECUTION

Program in Figure 7 explains the possible ways to run the program. The program's execution has different choices:

1. The direct execution: in which the program is translated to the machine's language and then executed on the machine's processor. Figure 8 presents the result of the direct execution of a program.
2. The cross-language translation: the DHAD's program is translated to C and/or Assembly languages. Figure 9 shows the translation of a program written in DHAD language to both C and Assembly languages. The execution can be done from any language after translating it to the machine's language as shown in Figure 10.
3. The execution of the translated program on the simulator. The execution of a program using the simulator is presented in Figure 11.

We chose to design and implement the assembler that translates a program written in assembly language to the machine's language on a PC rather than use a third-party assembler like MASM and ASM due to three main reasons:

1. This assembler can be used separately, and then, all messages like errors, warnings, and help are written in Arabic.
2. When translating the DHAD programming language program into assembly language, this assembler can be used to translate to machine code and then the program can be executed.
3. This assembler's code is incorporated into the back-end compiler to directly perform the translation from the source language to the target language.

Figure 7: Run menu

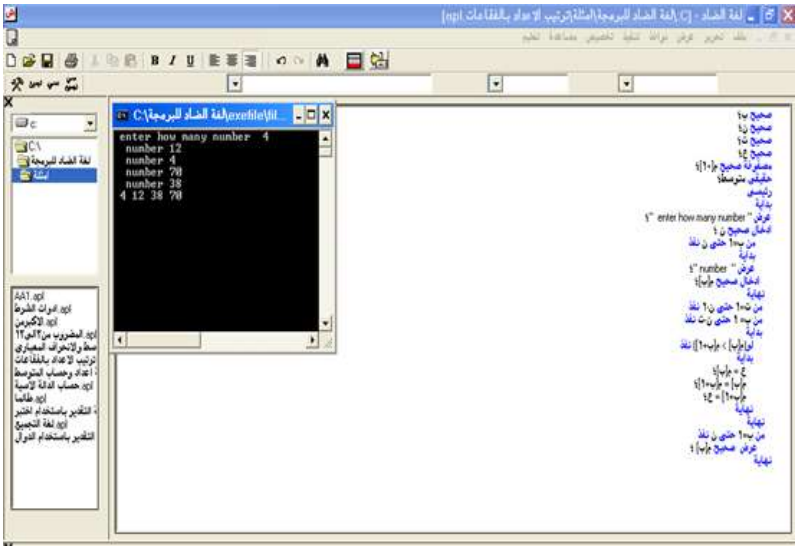

Figure 8: Direct execution

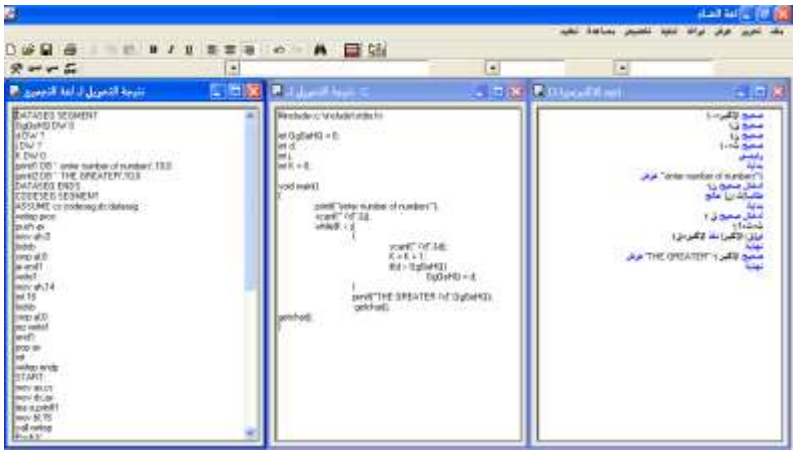

Figure 9: Translation to C and Assembly

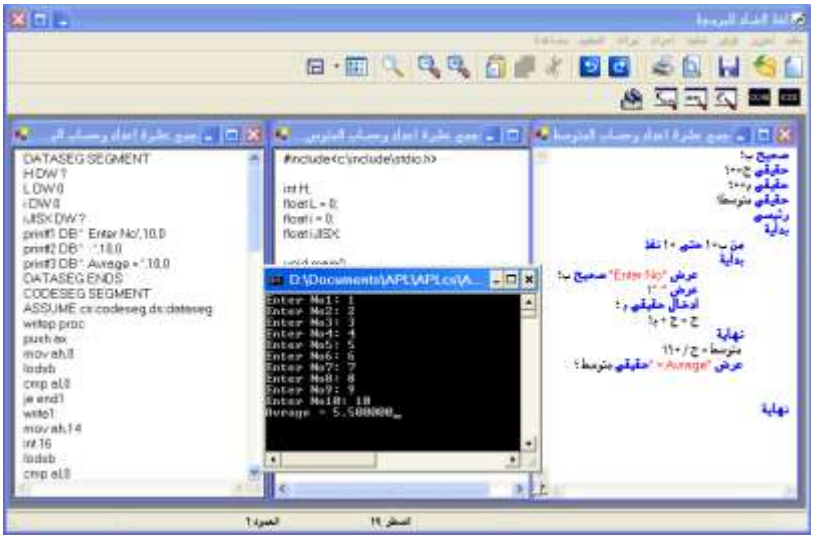

Figure 10: Execution from any program

The execution of the program can be on the machine or using the simulator as shown in Figure 11.

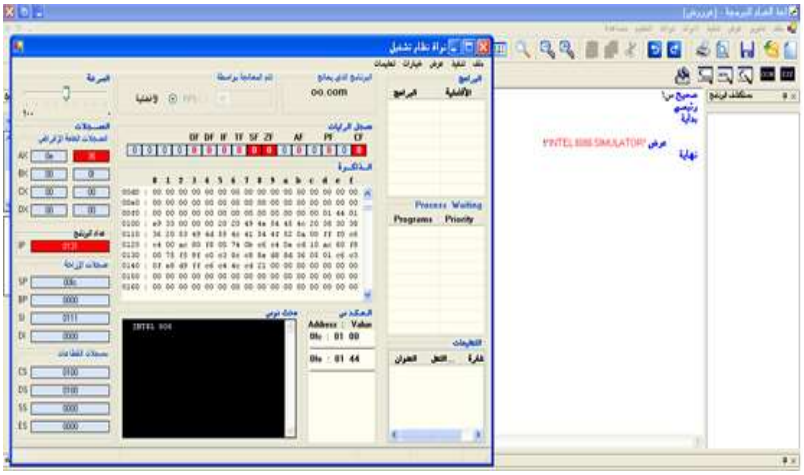

Figure 11: Execution using the simulator

The simulator helps the students understand the ways in which the machine executes a program. The user can control the execution step by step. Also a kernel is added to the simulator to manage the multiprogramming. Some settings such as the scheduling type and the speed of execution are specified by the user. All windows can be displayed or hidden separately. Figure 12 shows the process' transition states.

As soon as a new program is submitted it will be first placed on “hold” and then “ready state”. Depending on the scheduling type, the process control block (PCB) is put in its place in the list of processes. Once the current running process finishes its quota (or its execution) the CPU control is passed to the first process in the list. If the current running process has an input/output operation the process should be suspended, entering a waiting state. For simplicity, this state is omitted in the simulator and the time of input/output is reduced to a normal operation time.

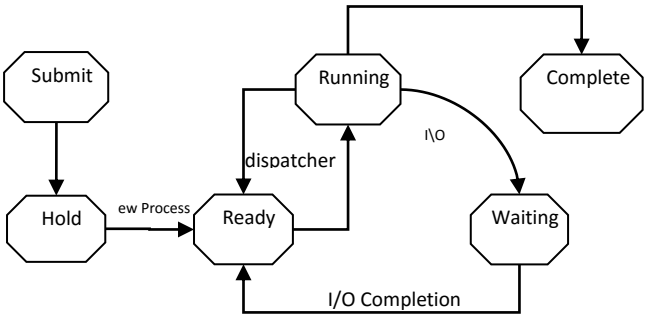

Figure 12: Process states

6. CONCLUSION AND FUTURE WORK

The aim of this project is to provide to the Arabic speaking students a way to self-learn different programming related courses. At this level, the tool, even if it is not completely achieved, can be used to provide an idea of the DHAD, Assembly and C programming languages. Also, the tool presents an overview of the machine’s architecture and how a program is executed over this architecture. The kernel explains how multiprogramming is managed. Future works will concentrate on how the students can be asked to modify the code to have their own grammar rules for the compiler and their own control modules in the operating system. The library and targeting exe file translation instead of com file will also be a focus.

7. ACKNOWLEDGMENT

The author would like to express his gratitude to the Deanship of Scientific Research’s support given under the project ID 3316. Also he would like to thank Abdulrahman H. Almotairi, Khaled N. Alotibi, Abdulqader Z. Almotairi, Saad N. Almotairi, Dakeel Allah Eid. A. Alolweet, Ali Abdullah S. Alolweet, Thamer Ali M. Alsayq, Suliman Abdullah Alsuhibany, Fahad Alnafesah, Ibraheem Alawad, Bader Massad Almotairi, Fahad Abdullah Alharbi, Mohamed Nayer Almotairi, Fares Sweleam Alrashidi, Mesheal Kalaf A. Alharbi for their work in the their final year projects to make this dream come true.

8. REFERENCES

Galperina, Marina, “Arabic Programming Language at Eyebeam: قلب Opens The World”, January 24, 2013
http://animalnewyork.com/2013/arabic-programming-language-at-eyebeam-%D9%82%D9%84%D8%A8-opens-the-world, Ramsey Nasser. /

Al-A'ali, Mansoor and Hamid, Mohammed. “Design of an Arabic programming language (ARABLAN), , Computer Languages, Volume 21 Issue 3-4, October, 1995, Pages 191-201, Pergamon Press, Inc. Tarrytown, NY, USA

Elsheikh, Mustafa “ARLOGO: The First Arabic Programming Language Project” (last retrieved 2014) http://arlogo.sourceforge.net

Amin, M.R., "The Arabic object-oriented programming language Al-Risalh," in Computer Systems and Applications, ACS/IEEE International Conference on. 2001, vol., no., pp.424-427, 2001 doi: 10.1109/AICCSA.2001.934031

Essam M. A., "Design of an Arabic object-oriented programming language and a help system for pedagogical purposes", 1996 Doctoral Dissertation, Illinois Institute of Technology Chicago, IL, USA.

Al-Ethawie, Jamal M. K. "Development of an Arabic programming language based on logic", Msc thesis, Al-Nahrain University, 1997.