PROGRAM ENVIRONMENT FOR INVESTIGATION OF MICRO-LEVEL COMPUTER PROCESSING

Radi Romansky *

Department of Informatics, Technical University of Sofia
Bulgaria

***** Corresponding author, e-mail: rrom@tu-sofia.bg

Abstract: The article deals with an approach for program description of procedures for information processing at a micro level in computer spice. The goal is to develop an example workspace with program modules for describing basic micro-operations which will allow conducting a program investigation of procedures for processor data processing. The TryAPL-2 operating program environment was used for this purpose. Developed models of basic nodes are presented, as well as the realizations of exemplary micro-operations.

Key words: mathematical formalization, discrete modelling, computer processing, micro operations, program tools.

1. INTRODUCTION

Firmware investigation and evaluation of its correct functionality is a significant problem in various areas of machine calculations, computer equipment and embedded systems. For example the general approach for firmware analysis is discussed in [1], directed to identification of firmware module, its extraction, disassembling, modification, and reprogramming, as well as dynamic and static analysis. This research area is particularly relevant in the development of embedded devices for the purposes of the Internet of Things (IoT), which is reflected in [2]. The article discusses the problem of analysis and verification of the firmware code used in embedded IoT device. Another point of view of the firmware, related to its storage in specialized memory, is discussed in [3]. This memory is a part of the computer equipment and it could be subject to external influences and possible attacks. A case of the analysis of compromised firmware is presented in [3], where a methods for identifying firmware-level threats in the memory forensic is proposed.

It is known that the organization of computer processing at the micro-level is based on the implementation of basic micro-operations to maintain machine calculations. Each machine program (firmware) consists of a number of microinstructions, which are executed in a certain order depending on logical conditions. Thus, several possible sequences of executed microinstructions (variants for realization of the machine algorithm) are formed in the microprogram. The optimization of machine calculations determines the need to study the temporal and structural parameters of firmware control, most of which are stochastic in nature. One possibility for formalization of the machine calculations is through a directed graph, in which the individual vertices represent the microinstructions, and the edges – the allowed transitions between them. In this case, the investigation is related to determining the possible paths from the beginning (input) to the end (output) of the firmware and calculating their lengths, as well as the probabilities for the realization of each of the possible paths under given initial conditions.

The goal of the article is to present a formal procedure and program modules for analysis and evaluation of firmware control of computations in micro-level. Proposed approach is based on the graph theory and operation research as the program realization of the models is made by using program language APL2 in the operational environment TryAPL2, which can be used for mathematical formalization of the computation processes and structures, including subjects in micro-level processing [4]. This is a parallel program language for processing of arrays with three basic components – definition of virtual vector (array) for storing items; definition of a relative storage address where the array could be stored; and definition of the virtual distance between subsequent elements [4].

The article has the following structure – next section discusses related work in the area, section 3 presents some features of the used program environment, section 4 proposes formal models of some basic structural element and micro-level procedures realized by using APL2, and finally some conclusions and future work is discussed in the last section.

2. RELATED WORK AND BACKGROUND

Firmware is an important part of the micro-level control of the computer processes and requires serious measures for organization, storage and maintenance depending on the specific application. In this regard, a method for evaluation of the source code for portable IoT device is discussed in [2]. The firmware is a part of the software program code and illegal modification is unacceptable. Measures for verification of processes on micro-level must be used to guarantee the correct functionality both in the preliminary organization and in the subsequent use in working conditions. Comparison of program codes for firmware validation is the approach proposed and discussed in [2]. Authors declare that "*In this case study, we analyzed and verified the object firmware code with the detailed item selection and suggest an additive comparison item in embedded IoT device*".

As mentioned in the previous section, the role of memory in firmware storage is discussed in [3]. The purpose of this article is to investigate the manner of the firmware code to be accessed over the memory bus and "*a survey of firmware rootkit* *techniques and the traces such threats leave in a system*" is made. Methods for examination of firmware components are developed and implemented in open-sources utilities with preliminary evaluation of its reliability with the help of a proof-of-concept rootkit.

The problem of firmware investigation is discussed in different directions aimed at model approach. For example, in [5] is presented a project for the development of an interactive educational simulator (called WepSIM) for study of firmware control in computer calculations, defining 3 main objectives: (1) ensuring stable interaction between hardware and software components of the system by designing a reliable set of microinstructions; (2) offering a universal interactive tool for low-level functionality investigation in a basic processor performing the set of microinstructions; (3) expanding students' knowledge through innovations in university courses. The article presents the results which are obtained during the carried out experiments by using this firmware simulator in the field of computer structures.

It is clear that each microinstruction defines an elementary (atomic) operation that must be performed when it is called by the firmware. In this regard, in [6] is a patented flexible system of microinstructions for design a working firmware to perform processes for servicing various applications and tasks. The goal is the designed firmware to be able to be compiled for execution using a suitable compiler in each program language, with providing flexibility of microinstructions in the implementation of various business processes.

An approach for organization of low-level model investigation is discussed in [7], proposing a unified framework for the joint conduct of the classification process and the modeling. The goal is to ensure the effectiveness of model study, and the developed model is common enough to include any type of modeling at the micro-level. Although the developed framework is aimed at image analysis, the approach provides basic guidelines for the organization of collaboration between a classifier (pre-extraction of variables) and the organization of low-level model research.

Testing the firmware that is embedded in various devices to control the processes at micro-level is limited by its strong dependence on the hardware structure and the poor scalability of the components. This is especially important for IoT devices used for monitoring and measuring the observed parameters. In addition to the above article [2], a software framework for the continuous execution of binary firmware together with independent testing has been proposed in [8]. The proposed P2IM technique is not affected by the peripherals (without ignoring it) and processes the input-output data for the firmware based on automatically generated models. The effectiveness of the framework was assessed using different firmware, and the results showed 79% of successfully completed tasks without manual assistance. In addition 7 unique unknown bugs were identified in the experiments.

An area that reflects up to the two articles mentioned above is the analysis of big data or of distributed data processing systems, which is discussed in [9]. The article proposes a data management scheme for modular calculations at the micro-level, looking for efficiency in accelerating the processing of images, text and sensory data. The authors confirm that various researches and relevant implementations have been conducted for data and metadata management, but justify their proposal with the fact that there are no specific investigation of the impact on data at the micro level and when using intermediate data which motivated them to propose this working scheme.

3. METHODS, TOOLS AND PROCEDURE

The program environment developing is based on principles of the graph theory and the Operation Research (OR) used to determine principles of the investigation and to form the conceptual models of the studied objects. The program realization is performed by using language APL2 in the operational environment TryAPL2. The developed program modules represent additional functions to the environment, allowing to increase its descriptive possibilities. Each of the created functions can be executed independently or as a series of calls. The system workspace OR (Operations Research) of the TryAPL2 environment was used to perform the experiments which support the functions presented in Table 1.

Table 1. Main supported functions by OR

Function	***Purpose***
SETUP *matrix*	Defines a specific STN (State Transition Network), described by a numerical matrix of connections matrix. Defines: *NODES* - list of node numbers; *SIZE* – number of nodes; *NETWORK* – a copy of the matrix; *CM* – connection matrix. To illustrate the functionality, the OR supports a matrix called SPM, describing the weight connections in a seven-node network.
PATHSFROM *node*	Calculates all paths in the STN from the specified *node* to the last node. In an STN with one input node and one final node, all paths are calculated by ρPATH←PATHSFROM 1
ARCS *path*	Determines the weight of each connection (arc) of the attribute *path*. Examples: ARCS ↑PATH – displays information about the first of the paths stored in PATH; +/ARCS ↑PATH – determines the length of this path; +/¨ARCS¨PATH – determines the length of all paths in STN.
VALUE *path*	Similar to +/ARCS *path*, where *path* may contain one (↑PATH), more or all (¨PATH) possible paths.

The mathematical formal model of a microprogram can be presented as a set of states (microinstructions) and possible transitions between them. In essence, this is an abstract graph model, presented as STN (State Transition Network), which uses a weight graph with probabilities for the realization of the corresponding transition along the edges. The individual transition probabilities are determined on the basis of a preliminary empirical analysis of the possibilities for different input data.

The proposed procedure is a sequence of abstract and technological operations for research and evaluation of firmware. An exemplary abstract microprogram was used in the formulation of the sequential actions (Fig. 1).

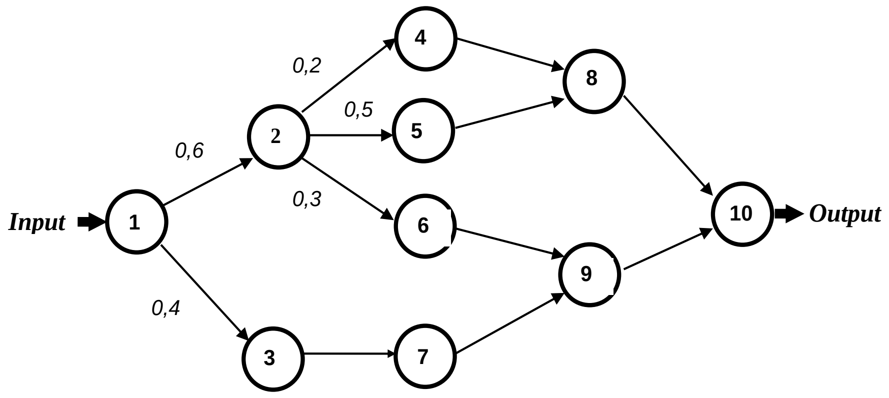

Figure 1. Abstract microprogram

It is envisaged that the individual micro-operations (nodes in the graph) will be realized by successive calls of standard and additionally developed functions depending on the probabilistic parameters of the transitions. The procedure includes the steps presented below.

1. Analysis of the microprogram (with N microinstructions) and definition of a formal abstract model as a directed weight graph representing STN with N states (in case of fig. 1 N=10) and value determination of the transient probabilities in the range [0, 1]. The graph formal model is built with one input and one output nodes, which corresponds to the structural organization of the firmware.

2. Defining data structure (matrix) with dimension *N*×*N* and entering the matrix of MATRIX connections in the APL2 environment (using the weights along the edges of the abstract graph model).

```
)LOAD OR
MATRIX←10 10ρ0
MATRIX[1;2]←0.6
MATRIX[1;3]←0.4
MATRIX[2;4]←0.2
MATRIX[2;5]←0.5
MATRIX[2;6]←0.3
MATRIX[3;7]←MATRIX[4;8]←MATRIX[5;8]←1
MATRIX[6;9]←MATRIX[7;9]←MATRIX[8;10]←MATRIX[9;10]←1
```

3. Initial quantitative analysis of the created formal description through the software of APL2 in the workspace OR – it is performed by the standard functions of Table 2. Upon execution of each of them, an answer related to the specific examination of the firmware is returned.

Table 2. Standard functions in OR for SETUP MATRIX

SETUP MATRIX	***Comment***
NODES	Determines the numbers of the nodes in the STN
SIZE	Displays dimensions of the connection matrix
NETWORK	Displays defined STN
CM	Displays the connection matrix defined by MATRIX

4. Determining the reachability between the initial and final node and storing all defined paths in an internal structured variable for further study with the possibility of visualization of a selected path - Table 3.

Table 3. Commands for investigation of reachability

Command	***Comment***
PATHSFROM 1	Determining possible paths from input (node 1) to the output
ρPATH←PATHSFROM 1	Stores all defined paths in the structural variable
↑PATH	Displays the first path in the structure PATH

5. Quantitative investigation of the weights of the possible paths based on the individual stochastic parameters of the individual nodes and calculation of probabilistic estimates for one or more paths in the structure – Table. 4.

Table 4. Determining the parameters of the paths

Command	***Comment***
ARCS ↑PATH	Determines the weights of the arcs for the first path in PATH
x/ARCS ↑PATH	Calculates the multiplication of the probabilities between nodes in the first path
A←ARCS¨PATH	Stores in A the weights (probabilities) for all determined paths
x/¨ARCS¨PATH	Determines the multiplication of all probabilities in each path in STN
VALUE¨PATH	Determines the sum of the probabilities for the each path in the STN
+/¨ARCS¨PATH	Similar to the previous one

6. Analysis of the probabilistic parameters for the investigated microprogram and definition of weight limit values (minimum / maximum) for reachability parameters – Table. 5. The latter are related to the time parameters for a specific execution of the firmware.

Table 5. Analysis of probability parameters

Command	***Comment***
V←x/¨ARCS¨PATH	Stores the result of the operation in a work variable V
MIN←⌈/V	Calculates the minimal probability multiplication for a path
MAX←⌊/V	Calculates the maximal probability multiplication for a path
(V=⌈/V)/PATH	Determines the path in the STN with maximum result of the probabilities multiplication

4. PROGRAM MODULES AN INVESTIGATION ORGANIZATION

The developed programming eenvironment contains various programming modules in the APL2 language environment for automation of firmware model research for low-level computer computing management. The preliminary staging is based on a mathematical formalization, adapted to the linguistic possibilities of the environment. The main purpose is to describe and model the study of machine algorithms for optimizing firmware in terms of memory used, execution time, data exchange with other firmware and ensuring the necessary accuracy of operations.

It is clear that atomic operations supported by microinstructions are realized in the basic structure of a processor device (mainly registers with different purposes - addresses, data, control, flags), which determines the use of data structuring (Boolean vectors of length equal to the machine word). This allows ensuring the adequacy of the model presentation and study of operations. The development of analytical models of sample micro-operations (machine algorithms) is based on basic elementary transformations supported in a hardware environment of a traditional processor, some of which are presented below (Figure 2).

a) ***INVERS*** – model of a digital code invertor of the binary code entering in the inputs;

b) ***DCOD*** – model of a code converter;

c) ***COMPR*** – model of a comparator for comparing input binary codes X and Y (with equal length) to forming output result Z on the principles Z = 0 (*if* X≥Y) or Z = 1 (*if* X <Y);

d) ***ADDER*** – model of a parallel adder for unsigned multi-digit binary numbers.

When organizing model experiments for the study of basic operations for computer processing at a low (micro) level, the following is performed:

1. Specifying the register structure for the operation.
2. Development of the machine algorithm.
3. Formalization of the machine algorithm for representation of the elementary operations in canonical mathematical form.
4. Creation of a program model of the formalized algorithm in APL2 environment.
5. Conducting test performances with the model.

```
     ∇INVERS  P; I
[1]    I←ρP
[2]   ET1:→(P[I]=1)/ET2
[3]    P[I]←1
[4]    →ET3
[5]   ET2:P[I]←0
[6]   ET3:I←I-1
[7]    →(I≥1)/ET1
[8]    'RESULT IS: '
[9]    P∇

     X←1 0 1 1 0 1 1 0
     INVERS  X
RESULT IS: 0 1 0 0 1 0 0 1
```

a)

```
     ∇Y←DCOD X
[1]    N←ρX
[2]    Y←X
[3]    →(X[1]=0)/OUT
[4]    Y←~X
[5]    Y[1]←1
[6]   ET:Y[N]←Y[N]+1
[7]    →(Y[N]=1)/OUT
[8]    Y[N]←0
[9]    N←N-1
[10]   →ET
[11]   OUT: 'OPERAND IS:' , X
[12]   'RESULT IS: ',  Y∇

     X←1 0 1 1 0 1 1 0
     Y←DCOD  X
OPERAND IS: 1 0 1 1 0 1 1 0
RESULT IS:  1 1 0 0 1 0 1 0
```

b)

```
     ∇Z←X COMPR Y
[1]   Z←I←0
[2] REP:I←I+1
[3]   →(I>ρX)/OUT
[4]   →(X[I]=Y[I])/REP
[5]   →(X[I]>Y[I])/OUT
[6]   Z←1
[7] OUT: 'IF X≥Y => Z=0; IF X<Y
=> Z=1'
[8]   ' FLAG  Z  IS: ', Z∇

     Z←10110110 COMPR 11001010
IF X≥Y => Z=0;  IF X<Y => Z=1
FLAG  Z  IS:  1
```

c)

```
     ∇ S←X ADDER Y; C
[1]   →((ρX)≠(ρY))/0
[2]   X←0,X
[3]   Y←0,Y
[4] LOOP:C←X^Y
[5]   X←(((~X)^Y)∨(X^(~Y)))
[6]   Y←(1↓C),0
[7]   →(C≠0)/LOOP
[8]   S←X∇
```

d)

Figure 2. Program modules for formalization of micro-operations

To illustrate the described approach two APL software models for study of machine operations are presented in the figure 3: (a) normalization of floating-point numbers (NORM function); (b) multiplication of fixed-point binary numbers by the method of lower digits (MULT function).

They are developed on the basis of traditional register structures for the implementation of the respective operations at a low level, and in the formalization of the algorithms the correspondence of an operator order with an elementary operation (micro-operation) is sought.

```
     ∇ M  NORM  P
[1]  D1: ⎕←‘ENTER START = 1’
[2]  START←⎕
[3]  →(START≠1)/D1
[4]  P1: START←0
[5]  P2: FINI←0
[6]  P3: ZERO← 0
[7]  P4: R←M
[8]  P5: S←P
[9]  D2: →((+/R)=0)/P8
[10] D3: →(R[2]=1)/P9
[11] P6: R←R[1],(2↓R),0
[12] P7: S←((ρS)ρ2) ⊤ ((2⊥S)-1)
[13] →D2
[14] P8: ZERO←1
[15] P9: FINI←1
[16] ‘RESULT  IS:’
[17] ‘FINI = ’, FINI
[18] ‘ZERO = ’, ZERO
[19] ‘M = ’, R
[20] ‘P = ’, S   ∇
```

Program execution:

```
M←1 0 0 0 1 1 0 0 1 0 1 1
P←0 1 1 0
M  NORM  P
```

(a) Normalization

```
     ∇ X  MULT  Y
[1]  D1: ⎕←‘ENTER START = 1’
[2]  START←⎕
[3]  →(START≠1)/D1
[4]  ⎕IO←0
[5]  P1: START←0
[6]  P2: FINI←0
[7]  P3: R1←X
[8]  P4: R2←Y
[9]  P5: S←8ρ0
[10] P6: B←7
[11] D2: →(R2[7]=0)/P8
[12] P7: S←(8ρ2) ⊤ ((2⊥S)+(2⊥(1↓R)))
[13] P8: R2←R2[0],0,R2[1 2 3 4 5 6]
[14] P9: R2[1]←S[7]
[15] P10: S←0,S[0 1 2 3 4 5 6]
[16] B←B-1
[17] D3: →(B≠0)/D2
[18] P11: S[0]←(R1[0]≠R2[0])
[19] P12: FINI←1
[20] ‘RESULT  IS:’
[21] ‘FINI = ’, FINI
[22] ‘X . Y = ’,  S,(1↓R2)
[23] ⎕IO←1   ∇
```

Program execution:

```
X←1 0 0 0 1 0 1 0
Y←0 0 0 0 0 0 1 0
X  MULT  Y
```

(b) Multiplication

Figure 3. Program APL2-models of basic machine algorithms (microprograms)

5. CONCLUSION AND FUTURE WORK

An approach for mathematical formalization and program realization of basic actions at micro level is proposed, as basic modules from the developed model space are presented. The procedures are fully compliant with the supported functions of the experimental environment APL2 and with the requirements for mutual communication between them when constructing firmware. Two basic machine operations are used to illustrate testing the operational capabilities and functionality of the approach.

REFERENCES

[1] TechPats. *Firmware Analysis*, Available at (visited in September 2020): https://www.techpats.com/technology/systems-and-software/firmware-analysis/

[2] Lee K., Lee H., Kwon S.Y., Nam S., Kim D. An Evaluation to Firmware Code Materials for Embedded IoT Device. In: *Hwang S., Tan S., Bien F. (eds) Proceedings of the Sixth International Conference on Green and Human Information Technology. ICGHIT 2018. Lecture Notes in Electrical Engineering*, vol. 502, 2019. pp. 227-231. Springer, Singapore. https://doi.org/10.1007/978-981-13-0311-1_38

[3] J. Stüttegen, St. Vömel, M. Denzel. Acquisition and analysis of compromised firmware using memory forensics. *Digital Investigation*, no. 1**,** vol. 12, March 2015, pp.550-560, http://doi.org/10.1016/j.diin.2015.01.010

[4] Engel, S. Writing circuit histories. *Fast Capitalism*, vol. 15, 2018, pp. 19-30; doi:10.32855/fcapital.201801.004

[5] F. García-Carballeira, A. Calderón-Mateos, S. Alonso-Monsalve and J. Prieto-Cepeda, "WepSIM: An Online Interactive Educational Simulator Integrating Microdesign, Microprogramming, and Assembly Language Programming," in *IEEE Transactions on Learning Technologies*, no. 1, vol. 13, 2020, pp. 211-218, doi: 10.1109/TLT.2019.2903714.

[6] Patricio Osvaldo Barletta, Julian Esevich Sanchez, Eduardo Adrian Cominguez. *Flexible microinstruction system for constructing microprograms which execute tasks, gateways, and events of BPMN models*. US Patent 10,037,197, 2018, https://patents.google.com/patent/US10037197B2/en

[7] Adrien Lagrange, Mathieu Fauvel, StéphaneMay, NicolasDobigeon. Hierarchical Bayesian image analysis: From low-level modeling to robust supervised learning. *Pattern Recognition*, vol. 85, Jan 2019, pp.26-36, https://doi.org/10.1016/j.patcog.2018.07.026

[8] Bo Feng, Alejandro Mera, and Long Lu, P2IM: Scalable and Hardware-independent Firmware Testing via Automatic Peripheral Interface Modeling. In *Proceedings of the 29th USENIX Security Symposium*, 2020; Available from https://www.usenix.org/system/files/sec20spring_feng_prepub_0.pdf

[9] Debasish Chakroborti, Banani Roy, Amit Mondal, Golam Mostaeen, Chanchal K. Roy, Kevin A. Schneider, Ralph Deters. A Data Management Scheme for Micro-Level Modular Computation-Intensive Programs in Big Data Platforms. *In: Alhajj R., Moshirpour M., Far B. (eds) Data Management and Analysis. Studies in Big Data*, vol. 65. 2020, Springer, Cham. https://doi.org/10.1007/978-3-030-32587-9_9

Information about the author:

Radi Romansky is a full professor at Technical University of Sofia, Department of Informatics, Ph.D. in Computer Engineering and D.Sc. in Informatics and Computer Science; Full member of European Network of Excellence on High Performance and Embedded Architectures and Compilation (HiPEAC). He has over 200 scientific publications and over 20 books. Areas of scientific interests: ICT, informatics, computer architectures, computer modelling, data protection, etc.

Manuscript received on 07 December 2020