Instruction Prediction in Microprocessor Unit

Andrzej Kwiecień, Michał Maćkowski, and Krzysztof Skoroniak

Silesian University of Technology, Institute of Computer Science,
Akademicka 16, 44-100 Gliwice, Poland
{akwiecien,michal.mackowski,krzysztof.skoroniak}@polsl.pl
http://www.polsl.pl/

Abstract. Protection of computer systems from an unauthorized access to the classified information is a very essential issue. The research deals with some aspect of this problem resulting from the fact, that for instance, an author of a software for embedded system is not aware that it is possible to identify partly or entirely, program code. It can be done in non-invasive way, without influence on the microprocessor internal structure and program memory. The authors intend to prove that it is possible to recognise the instructions executing by certain type of microprocessor, analysing only the character of disturbances in the power supply lines. The research results inspire to the more careful study of ways and methodology for developing software, which should highly hinder the software reverse engineering.

Keywords: reverse engineering, program code, microcontroller, conducted emission, electromagnetic disturbances, electromagnetic interference.

1 Introduction

The fast expansion of microelectronic in various fields of technology, which is the result of rapid technological development, caused that more and more people deal with the exploitation and development of devices based on a microprocessor unit. The maximum of costs, during the construction of devices based on a microprocessor unit, is related to system design and time spent on writing software which would execute the assumptions presented to the programmers. In this case, the manufacturer must be aware of the security of its product and program contained in the memory of microprocessor.

However, the lack of information about the security offered by a particular microprocessor is the reason for the wrong choice of the central processor unit which is not dictated by a security of a system, but a low price. Generally, each vendor of microprocessors can offer some methods of securing the program code and data from being accessed, in order to protect information. Though, there is no information on the level of such security and tests proving their effectiveness. In this situation, the manufacturer of a particular device should be aware of threats, which may arise when choosing the microprocessor, including the loss of classified information such as program code and encryption key.

A. Kwiecień, P. Gaj, and P. Stera (Eds.): CN 2011, CCIS 160, pp. 427–433, 2011.

Obtaining information about the operation of a device through the influence of its work or monitoring the parameters of its activity, is called side-channel attack. The existence of the "side" channel through which such information is obtained, is usually unintended and results from the construction of a device or technology, in which it was built. An example of this situation can be any electrical powered device – in this case electromagnetic signals result from the currents flow and existing voltages. Signals used to transfer information from one point to another by conduction or radiation in the form of electromagnetic waveforms, in a deliberate manner, create the transmission channel. On the other hand, the situation when the signals are unintentional and, what is more, carry the information about the state of device operation, refers to side channel and information passing. Emission of such signals is often called compromising emanation. Thus, all unintentional signals that can reveal information in the case of capturing and analyzing them, are considered as compromising emanation. The source of these signals can be any electrical device used to transmit, receive, store or process information. For example, in this case the network controller placed in the network card can be also considered as a microprocessor unit, which is responsible for data processing of transmitted frames.

Methods of analysis of the microprocessor program code based on registration of power supply changes presented in this paper, use the conducted compromising emanation for reconstructing processes occurring in the microprocessor. Such propagation of conducted signals is the result of instantaneous changes in current consumption from the power source, depending on the currently use of elements constituting the central unit [1,2].

2 Test Bench and Research Procedure

The test bench used in the research was presented and discussed in the authors paper *The Analysis of Microprocessor Instruction Cycle* [3]. That paper presents the ways of measurement of power supply disturbances and describe the elements constituting the test bench. The analysis of microprocessor program code on the basis of measurement of power supply changes, requires the knowledge on microprocessor architecture, instruction cycle organization and microprocessor instruction list.

The 8 bit processor PIC16F84A used in the research, is one of the very popular Mid-Range processors. Because of the fact that the simplest PIC16 family of processors is devoid of advanced peripheral blocks, authors could focus mainly on the analysis of microprocessor program code based on the measurement of power supply changes, without interfering into the construction and operation of peripheral circuit. PIC16F84A processor is made in the Harvard architecture, i.e. it has a separated data and program memory, and is characterized by a reduced instruction set RISC (Reduced Instruction Set Computers). The processor instruction list includes 35 commands, most of which are executed in one cycle. The exceptions are several jump and call subroutine instructions that are executed in two instruction cycles.

Microcontroller PIC16F84A instruction list contains:

- 1-cycle instructions (2 μs):ADDWF, ANDWF, CLRF, CLRW, COMF, DECF, INCF, IORWF, MOVF, MOVWF, NOP, RLF, RRF, SUBWF, SWAPF, XORWF, BCF, BSF, ADDLW, ANDLW, CLRWDT, IORLW, MOVLW, SLEEP, SUBLW, XORLW,
- 2-cycle instructions (4 μs): DECFSZ (1 or 2 cycles), INCFSZ (1 or 2 cycles), BTFSC (1 or 2 cycles), BTFSS (1 or 2 cycles), CALL, GOTO, RETFIE, RETLW, RETURN.

High performance of the used processor results from using instruction pipeline and 14-bit wide operation code, which, apart from instruction code can also include the argument. The microprocessor PIC16F84A used in the research has one executive stream, which means that during one command cycle only one instruction is realized. One instruction cycle (16 μs) consists of four machine cycles (4 μs):

- Q1 – instruction decode cycle,
- Q2 – instruction read data cycle,
- Q3 – process the data,
- Q4 – instruction write data cycle and fetching the next instruction from the program memory.

Due to the fact that in the last cycle (Q4) another instruction is fetched from the program memory, thus not only currently realized instruction but also next instruction has the influence on the current flow. In the research presented in this work all 1 and 2 cycles instructions were taking into account.

The way of receiving (writing) the instructions executed by microprocessor unit and the program structure during the test was presented in the previous authors paper [3] in the research procedure. In this paper the authors consider the completely different problem, that is they intend to predict the currently executed instruction on the basis of voltage disturbances in the power supply lines. To achieve these goals, first it is necessary to write each of the instructions in a certain way into the memory (Fig. 1). Thus, a database of instructions samples is created, which can be then compared with various instructions of test program (Fig. 2).

After determining the time waveform of processor voltage supply during the whole program operation, a part of the time waveform of a tested instruction is excised. Then, the minimum and maximum value of voltage for each machine cycle is saved – this includes six values that characterize a particular instruction. As it was mentioned when discussing the test bench, during the fourth machine cycle not only the result is written, but also another instruction is fetched from the memory. Therefore, the fourth machine cycle was not taken into consideration, because the address of another instruction in the memory influences also on the shape and level of voltage in this cycle.

In the previous research [4], each instruction was written with the use of amplitude spectrum designated for the voltage waveform while executing the

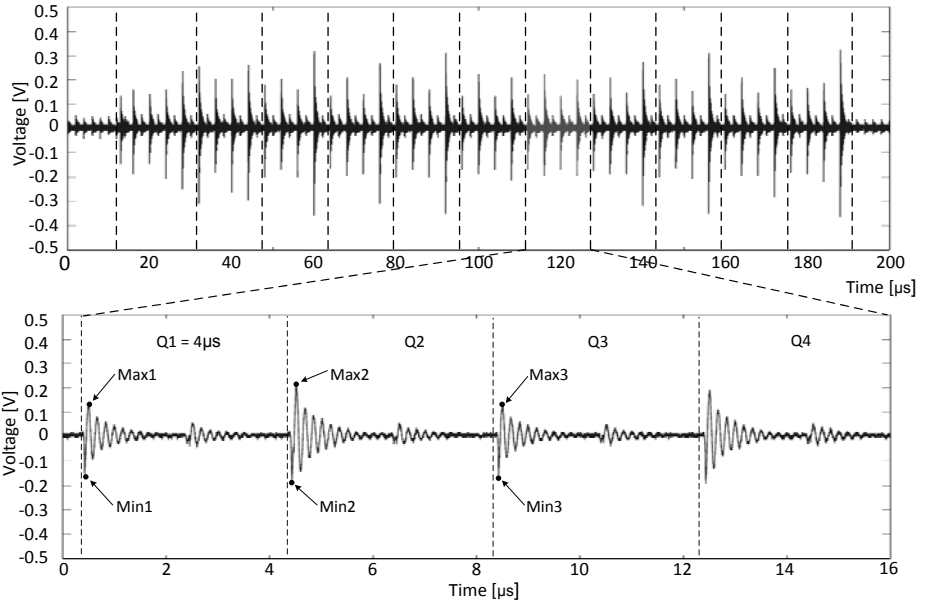

Fig. 1. The research procedure – microcontroller test program construction

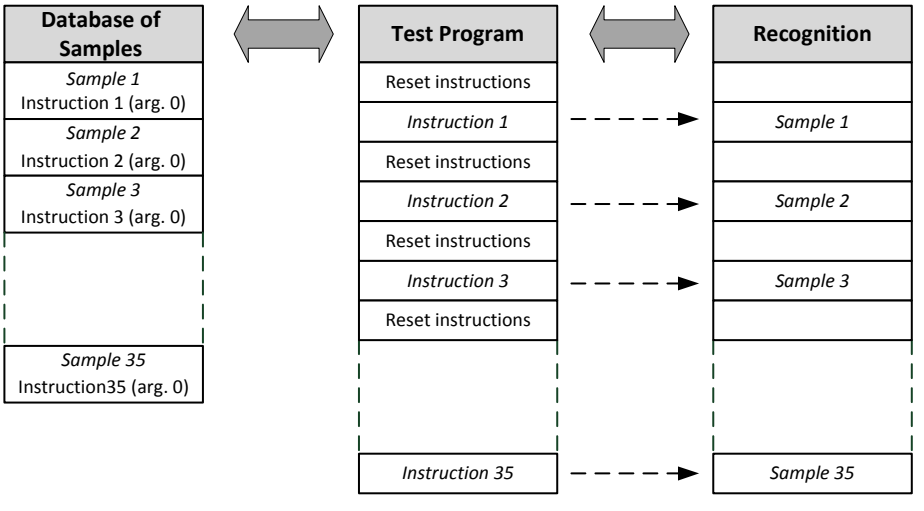

Fig. 2. The process of instruction identification

instruction. However, the method of writing and comparing the instructions which applies the minimum and maximum value, developed by authors and presented in this paper, gives better effects.

Having created the database of samples, the authors then wrote a test program, used to determine the effectiveness of the method (Fig. 2). The program consisted of individual instructions separated by instructions that reset the register – the database included also instructions operating on arguments with zero value. Then, each of the instructions of a test program (in the form of 6 minimum and maximum values) was compared to all samples in database using the method of the least squares. It enabled to reveal that a sample from the database, which was the most similar to the tested instruction, was then typed as a recognized instruction.

3 The Research Results

According to the research procedure, each instruction of a tested program was compared to all samples included in the database. The first column of Table 1 presents the following instructions of a test program that consists of all 35 microprocessor instructions. The following columns illustrate 6 instructions, which obtained the minimum values, using the least squared method.

Because of the fact that the least squares method was used, thus the smaller value this method returns (the number in brackets in Table 1), the more relationship is between instruction in a test program and instruction in database. In consequence of using presented research method, 32 of 35 instructions were recognised correctly – the proper equivalents of instructions were marked in grey.

The instructions are considered to be correctly recognised, only when their equivalents in database of the samples was found at the first step (column 2 in Table 1). For three instructions the corresponding samples in database were found in the further place. This refers to COMF, SWAPF, and XORWF instructions. In case of COMF instruction (complementary operation) using the least squares method, instructions DECFSZ and DECF were returned at the first step (both of these instructions realize the argument decrementing). As it was previously mentioned, instructions in the test program as well as those saved in the database of samples, operated on arguments with zero value. In this situation the operation of COMF and DECF is similar and leads to change of value of 8-bit argument from 0 up to 255. The negation of all 8 bits of argument in case of these two instructions causes, that it is necessary to switch a certain number of gates inside the microprocessor unit, which results, in this case, in rapid and similar current consumption from the power supply – this may explain incorrect instruction recognition.

4 Conclusion

This paper presents the method developed by authors, which determines the currently executing instruction based on the disturbances of the voltage supply.

Table 1. The results of instruction recognise for the following instruction of tested program

The following instructions of test program		RECOGNISED INSTRUCTIONS					
		Instruction 1 (result of least squares method)	**Instruction 2** (result of least squares method)	**Instruction 3** (result of least squares method)	**Instruction 4** (result of least squares method)	**Instruction 5** (result of least squares method)	**Instruction 6** (result of least squares method)
1	ADDWF	**ADDWF** (9.5)	**RLF** (12.2)	**XORWF** (13.0)	**SWAPF** (13.0)	**RRF** (15.2)	**MOVWF** (15.6)
2	ANDWF	**ANDWF** (2.2)	**BTFSC** (3.0)	**BTFSS** (7.4)	**MOVWF** (50.3)	**XORWF** (67.7)	**BCF** (70.3)
3	CLRF 15	**CLRF** (6.1)	**MOVWF** (8.7)	**RLF** (31.2)	**SWAPF** (31.2)	**ADDWF** (34.7)	**RRF** (43.0)
4	CLRW	**CLRW** (5.2)	**SUBWF** (109.4)	**BCF** (154.1)	**BSF** (220.5)	**ANDWF** (224.8)	**BTFSC** (245.7)
5	COMF	**DECFSZ** (5.2)	**DECF** (5.6)	**COMF** (11.7)	**RETURN** (6492.2)	**RETFIE** (6671.9)	**BSF** (7039.1)
6	DECF	**DECF** (10.0)	**DECFSZ** (12.2)	**COMF** (23.0)	**RETURN** (6549.2)	**RETFIE** (6728.3)	**BSF** (7072.0)
7	DECFZ	**DECFSZ** (10.0)	**DECF** (10.4)	**COMF** (17.4)	**RETURN** (6401.5)	**RETFIE** (6576.8)	**BSF** (6920.6)
8	INCF	**INCF** (3.5)	**INCFSZ** (16.5)	**BSF** (43.8)	**CLRF** (118.9)	**CLRWDT** (132.8)	**XORWF** (134.5)
9	INCFZ	**INCFSZ** (10.4)	**INCF** (11.3)	**BSF** (34.3)	**CLRF** (106.8)	**CLRWDT** (133.7)	**MOVWF** (147.6)
10	IORWF	**IORWF** (1.3)	**MOVF** (4.3)	**XORWF** (6.1)	**RRF** (10.0)	**ADDWF** (11.3)	**SWAPF** (22.6)
11	MOVF	**MOVF** (3.5)	**IORWF** (4.8)	**XORWF** (5.2)	**ADDWF** (6.9)	**RRF** (8.2)	**SWAPF** (14.8)
12	MOVWF	**MOVWF** (2.2)	**CLRF** (14.3)	**ADDWF** (16.9)	**SWAPF** (20.4)	**RLF** (21.3)	**RRF** (21.7)
13	NOP	**NOP** (5.2)	**IORLW** (78.1)	**ADDLW** (81.6)	**XORLW** (91.6)	**CLRWDT** (95.9)	**ANDLW** (108.1)
14	RLF	**RLF** (0.9)	**ADDWF** (3.5)	**RRF** (4.8)	**SWAPF** (6.9)	**MOVWF** (9.5)	**MOVF** (13.9)
15	RRF	**RRF** (1.7)	**ADDWF** (3.9)	**RLF** (4.8)	**MOVF** (6.5)	**IORWF** (8.7)	**MOVWF** (12.6)
16	SUBWF	**SUBWF** (3.0)	**BCF** (92.9)	**CLRW** (110.7)	**ANDWF** (115.0)	**BTFSC** (121.1)	**BTFSS** (136.7)
17	SWAPF	**RRF** (0.9)	**ADDWF** (1.3)	**RLF** (3.0)	**MOVF** (4.8)	**SWAPF** (6.9)	**IORWF** (7.4)
18	XORWF	**MOVF** (1.7)	**RRF** (2.2)	**IORWF** (3.0)	**ADDWF** (3.5)	**RLF** (9.5)	**XORWF** (9.5)
19	BCF	**BCF** (10.4)	**BTFSC** (61.2)	**SUBWF** (63.8)	**BTFSS** (64.7)	**ANDWF** (72.5)	**GOTO** (141.5)
20	BSF	**BSF** (13.0)	**INCFSZ** (13.5)	**INCF** (27.3)	**CLRF** (85.5)	**MOVWF** (133.2)	**RLF** (161.9)
21	BTFSC	**BTFSC** (10.9)	**ANDWF** (13.5)	**BTFSS** (19.5)	**MOVWF** (55.6)	**BCF** (59.0)	**XORWF** (63.4)
22	BTFSS	**BTFSS** (5.6)	**BTFSC** (6.5)	**ANDWF** (8.2)	**BCF** (53.8)	**MOVWF** (54.7)	**XORWF** (76.4)
23	ADDLW	**ADDLW** (13.9)	**XORLW** (18.7)	**IORLW** (27.8)	**MOVLW** (49.0)	**ANDLW** (54.3)	**NOP** (78.1)
24	ANDLW	**ANDLW** (6.5)	**XORLW** (64.7)	**ADDLW** (65.1)	**IORLW** (66.0)	**SUBLW** (74.7)	**NOP** (85.9)
25	CALL	**CALL** (7.8)	**SUBWF** (588.5)	**GOTO** (685.3)	**CLRW** (798.6)	**ANDWF** (947.0)	**BTFSC** (968.8)
26	CLRWDT	**CLRWDT** (1.3)	**IORWF** (36.5)	**XORWF** (51.6)	**MOVF** (51.6)	**RRF** (53.8)	**ADDWF** (57.7)
27	GOTO	**GOTO** (4.8)	**BTFSC** (145.0)	**BCF** (150.6)	**BTFSS** (154.5)	**SUBWF** (162.3)	**ANDWF** (167.5)
28	IORLW	**IORLW** (2.6)	**XORLW** (3.9)	**ADDLW** (6.1)	**MOVLW** (15.2)	**ANDLW** (58.6)	**NOP** (76.4)
29	MOVLW	**MOVLW** (7.4)	**IORLW** (8.7)	**XORLW** (16.1)	**ADDLW** (27.8)	**NOP** (112.8)	**ANDLW** (119.4)
30	RETFIE	**RETFIE** (9.5)	**RETURN** (11.3)	**SLEEP** (183.2)	**INCF** (204.4)	**RETLW** (208.3)	**INCFSZ** (247.8)
31	RETLW	**RETLW** (5.6)	**SLEEP** (76.0)	**RETURN** (135.0)	**RETFIE** (148.9)	**CLRWDT** (278.6)	**INCF** (289.9)
32	RETURN	**RETURN** (10.4)	**RETFIE** (11.3)	**INCF** (195.7)	**SLEEP** (204.9)	**RETLW** (223.1)	**INCFSZ** (249.6)
33	SLEEP	**SLEEP** (11.7)	**RETLW** (56.9)	**RETFIE** (147.1)	**RETURN** (154.1)	**INCF** (232.6)	**INCFSZ** (238.7)
34	SUBLW	**SUBLW** (6.1)	**MOVF** (54.3)	**BTFSS** (59.9)	**IORWF** (60.8)	**BTFSC** (66.8)	**ANDWF** (67.7)
35	XORLW	**XORLW** (1.7)	**IORLW** (4.8)	**ADDLW** (4.8)	**MOVLW** (11.3)	**ANDLW** (69.4)	**NOP** (89.8)

The methodology and conducted research that enable to determine and compare the spectrum of amplitude of a particular instruction were presented in the previous papers of authors [5,4]. The research conducted at that time was limited only to a one cycle instructions, and the result of a correctly recognised instructions was at the level of 80%. In this paper the authors focused mainly on improving the effectiveness of proper instructions recognition, and conducting the research for the entire microprocessor instruction list.

Enhancement of accuracy is achieved by placing the test bench in a shielded area (GTEM cell) so the test area was separated from the influence of external interference. The research procedure was also modified by changing the analysis of voltage disturbances in power supply lines in frequency domain into time domain. In consequence, this approach improved the instruction recognition.

In the conducted research it was possible to foreseen the currently executing instruction with the 91%. According to the research, 32 of 35 instructions for which the smallest value was obtained in the method of the least squares, have been correctly recognised. In the further research the authors intend to focus on developing the presented research method, in which also the instruction arguments will be considered.

References

1. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis: leaking secrets. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, p. 388. Springer, Heidelberg (1999)
2. Piotrowski, R., Szczepański, S.: Input gate current uses to differential power analysis cryptographic device. XI International PhD Workshop. Wisła (2009)
3. Kwiecień, A., Maćkowski, M., Skoroniak, K.: The Analysis of Microprocessor Instruction Cycle. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2011. CCIS, vol. 160, pp. 427–433. Springer, Heidelberg (2011)
4. Maćkowski, M., Skoroniak, K.: Instruction prediction in microprocessor unit based on power supply line. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2010. CCIS, vol. 79, pp. 173–182. Springer, Heidelberg (2010)
5. Maćkowski, M., Skoroniak, K.: Electromagnetic emission measurement of microprocessor units. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2009. CCIS, vol. 39, pp. 103–110. Springer, Heidelberg (2009)