A Training Software Model of an Interrupt System

Anelia Vasileva, Angel Smrikarov

Abstract: *The paper justifies the necessity to introduce the students from the 'Computer Systems and Technologies' degree course to the structure and way of operation of the interrupt system – one of the important components of the processor. Analysis of the basic functionality of an example interrupt system is presented, an existing interrupt system is selected as a prototype of the training model and the arguments for its selection are proposed. The paper also describes the implemented model and its features. The work with the model will enable students to comprehend the way of operation of the interrupt system and it will be also used to check and assess their knowledge.*

Key words: *Virtual Laboratory, Training Software Model, Simulation, Interrupt System.*

INTRODUCTION

The interrupt system is a basic component of each processor. In general, it could be said that the main task of the processor can be considered as selecting and handling the interrupt requests and that each currently executed program is actually a concrete interrupt handling routine, except the system initialization program. This makes the role of the interrupt system exceptionally important and its initialization – a very responsible task. The functioning of the whole computer system largely depends on the correct solution of this problem.

From the above presented it follows that the future specialists on computer systems and technologies should be familiarized with the structure and way of operation of the modern processors' interrupt systems.

One of the ways for achieving this goal is including a software training model of an Interrupt system in the virtual laboratory on "Computer organization" [1], a basic course in the curricula of a Bachelor in Computer Systems and Technologies.

The first step when designing a model of this type is to solve the following tasks:

1. Specifying the basic functionality of the interrupt system.
2. Specifying the developer's activity for initialization of the system.
3. Selecting an existing interrupt system as a prototype of the future training model.

The basic functions of the interrupt system [2] can be reduced to:

- defining the active edges or levels of the signals, accepted as interrupt requests;
- defining the priority of each one of the interrupt requests;
- enabling / disabling of separate requests;
- saving the incoming interrupt requests;
- saving the current processor status;
- switching the control to the interrupt handler routine;
- restoring the processor status after completion of the handling routine.

In general, interrupts are caused by events, which are external to the central processor and require immediate processing. Interrupts are provided primarily as a way to improve processing efficiency – for example when the processor communicates with external devices, which are much slower than it [3]. To ensure faster and more effective interrupt requests handling, the modern processors support interrupt vectors that define the initial address of each one of the interrupt handler routines and allow direct switch of the control to the corresponding one. In addition, assigning a priority to the interrupt requests guarantees the execution of the most urgent one in case several requests come simultaneously.

The initialization of the interrupt system is reduced to loading constants, specified by the programmer, in the following registers:

- the interrupt requests register and particularly in the bits which determine if the correspondent flag will be set by an edge or level;
- the interrupt priority register;
- the interrupt enable register.

The processors Pentium III and Pentium 4 with Intel IA-32 architecture [4] support an interrupt system that handles 17 predefined interrupts and 224 user defined, called maskable interrupts, separated in three groups: external hardware interrupts, internal hardware interrupts and exceptions or software interrupts. It is obvious that because of its complexity this interrupt system is not the appropriate prototype for a training model, targeted to fourth-semester students.

The interrupt system of 89C51 – one of the basic single-chip microcontrollers of MSC51 series [5] receives and handles 5 interrupt requests – 2 external and 3 internal (from the timer/counters and the serial port), as for each one an interrupt vector is supported at the beginning of the microcontroller's program memory. These requests are hardware, but could be software simulated. The receipt of an interrupt request causes triggering the corresponding flag – in other words – setting one of the triggers of the 8-bit registers TCON and SCON. Enabling / disabling and priority assigning of the requests is provided by setting the triggers of the 8-bit registers IE and IP. Figure 1 shows that this Interrupt system has comparatively simple structure and this makes it an appropriate prototype for the training model.

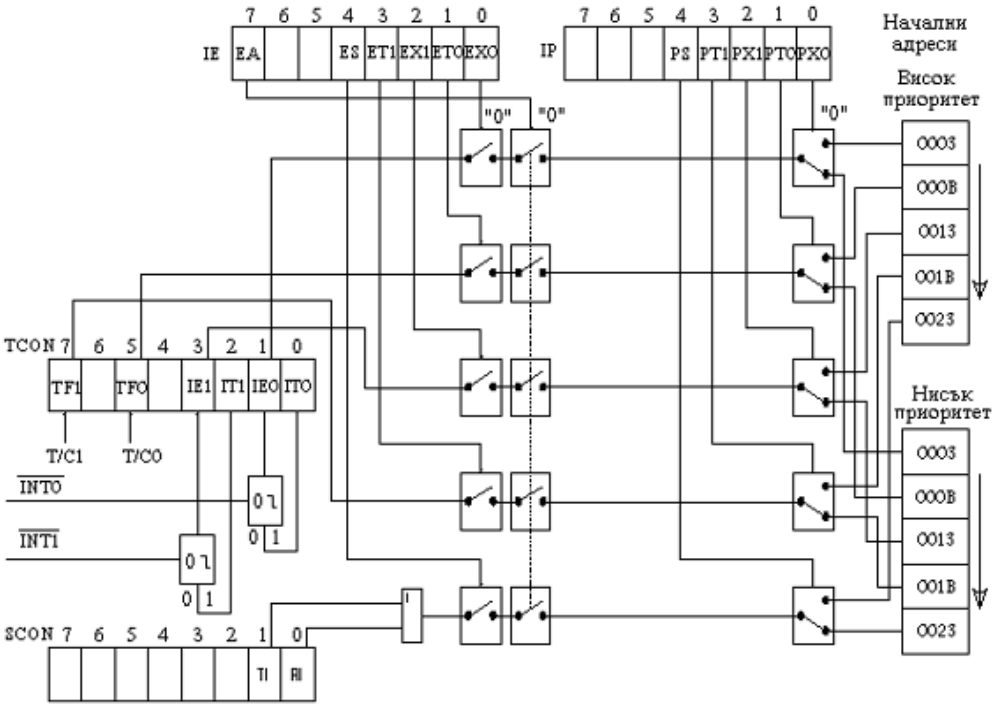

Figure1 Structural diagram of the interrupt system of 89C51

Another circumstance supporting the selection of 89C51 interrupt system as a prototype is the fact that the course "Computer organization" has links to the "Single-chip microcontrollers" considering the MCS51 series.

LAYOUT

In order to be effective as a learning tool and to be compatible with the other models in the Virtual Laboratory on Computer organization, the training model of the interrupt system should meet the requirements, specified in [1] and in terms of user interface, to be conformable with the basic principles of Human Computer Interaction (HCI). Not only should the model illustrate the structure of the interrupt system, but it should also enable the generation of concrete tasks for initialization, executed by the learners through a sequence of interactive actions. Besides, the model should enable the simulation of incoming interrupt request and control of learner's knowledge about the order of processor's actions, performed during the interrupt requests handling.

The analysis of the interrupt system of the single-chip microcontrollers of MSC51 series allows drawing the conclusion that the training model should consist of interrupt request register (IR), whose bits will be set when the corresponding interrupt request has arrived, interrupt priority register (IP) and interrupt enable register (IE) for assigning a priority to each one of the interrupt requests and their enabling / disabling. The external interrupt requests should be received by level or by edge of the incoming signal and this should depend on the content of two of the bits of IR.

The implemented model is a Windows' 9x, 2000, XP, NT application of size about 500KB, developed using the visual programming environment Delphi 5.

When starting the program, a window containing the structural diagram of the interrupt system (figure 2), and another one with randomly generated tasks for its initialization (figure 3), open on the screen. The second window remains active on the top until completion of the first stage of work with the model, including familiarizing with the tasks and execution of the necessary actions for the initialization of the interrupt system.

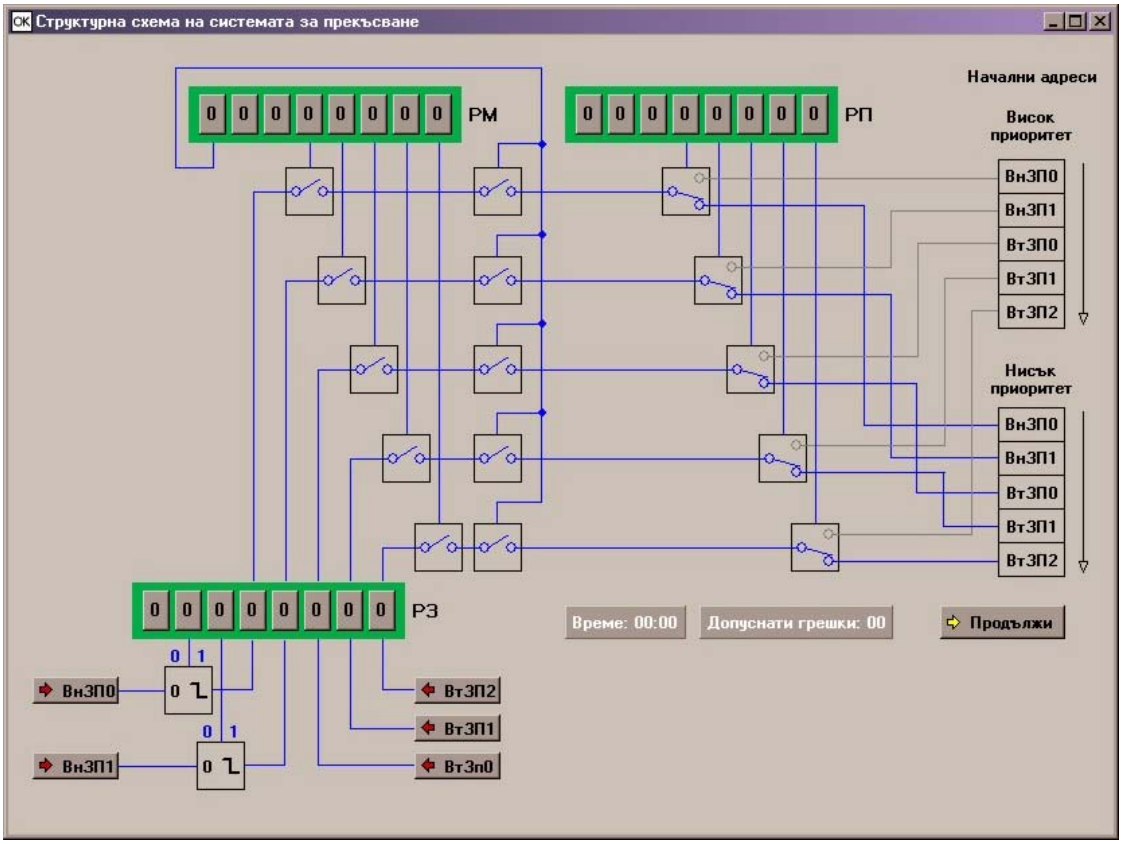

Figure2 A window containing the structural diagram of the interrupt system

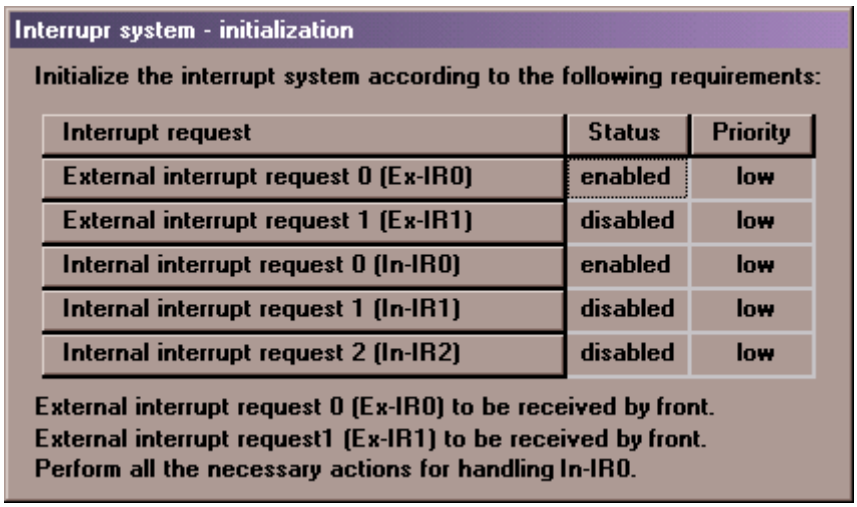

Figure3 A window with tasks for initialization of the interrupt system

By default all requests are disabled and low priority assigned, as IE and IP registers are reset. The learner has to enable the requests, pointed in the task window by clicking on the corresponding bit of IE. On wrong action a warning message appears and the mistake is counted out in the field "Mistakes". On a consecutive wrong action, the mistake is counted out and a message, hinting the correct action appears. Clicking on a bit of IE register sets it and turns the corresponding switch on, and clicking on the highest bit – "enable all interrupt requests" turns all the switches, connected to it, on. A correct click on a bit of IP register sets it and the corresponding switch position changes from "Low" to "High" priority, the connection to the low priority vector table becomes inactive and the connection to the high priority vector table becomes active. When one of the external interrupt requests, according to the assigned task, has to be received by edge, the learner has to "set" the appropriate bit of IR register by clicking on it.

Only if the interrupt system initialization is completed, the learner can "send" the request, specified in the task window, by clicking the appropriate button. After submitting the request, a button "Continue" appears. Clicking the button causes opening a window (fig. 4) where the learner has to order in a correct sequence all the actions, performed by the processor during the submitted interrupt request handling.

The list of actions appears in the right side of the window. The sequence of items is randomly generated and changes on each new window opening. The learner has to fill in the empty fields via drag-and-drop operations. When attempting to put a wrong list item in a field, a warning message appears and the text returns on its previous position. This part of the model does not support messages, hinting the correct action. After filling all the fields with the correct sequence of actions, an "End" button appears. Clicking on the button closes the window and generates a message containing information about the total work time with the model and the number of mistakes. Finally the program is automatically set in initial status and a new task is assigned to the learner.

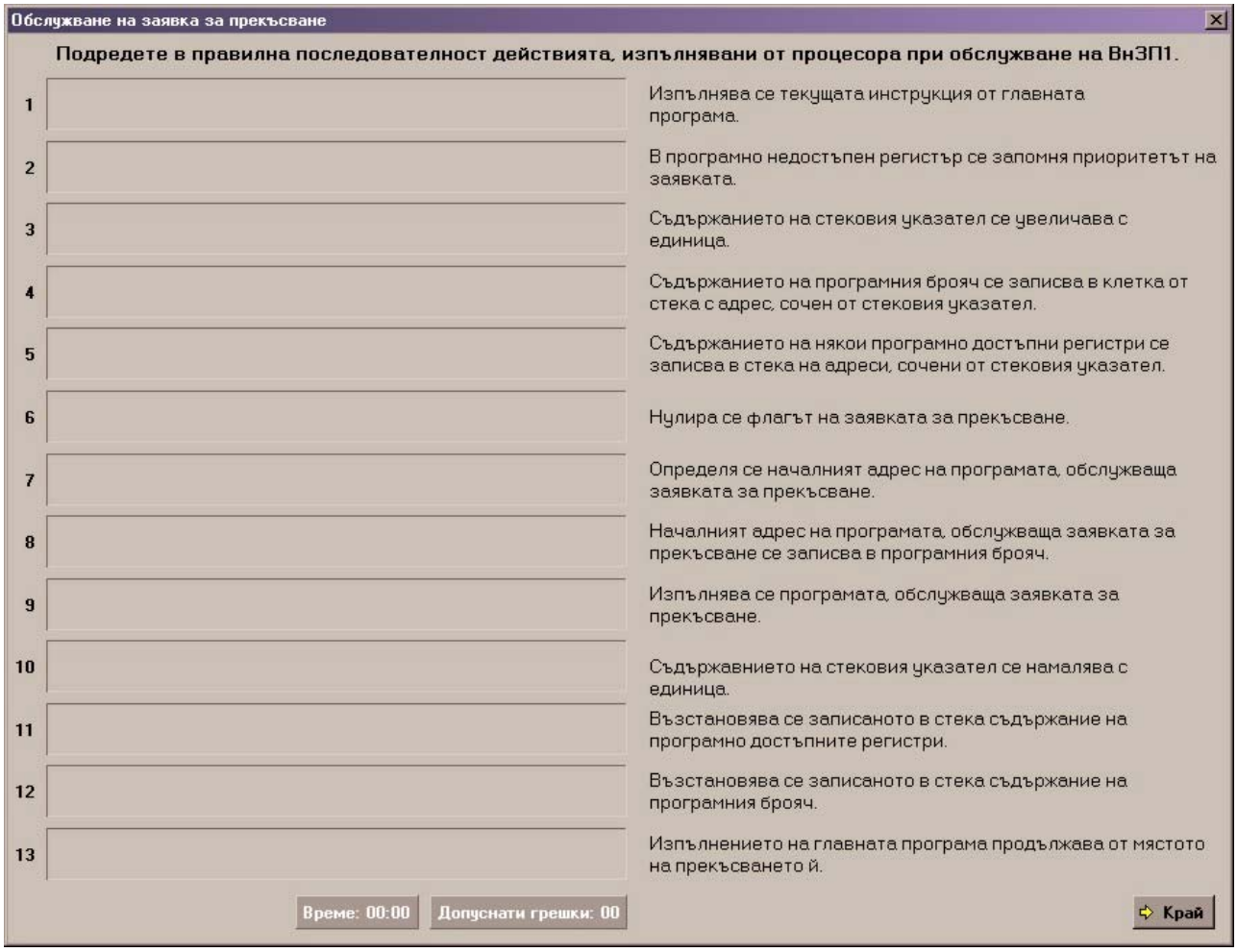

Figure4 A window with a list of actions, necessary for interrupt request handling

CONCLUSIONS AND FUTURE WORK

The implemented software training model gives an opportunity to illustrate the structure and way of operation of an exemplary interrupt system – from the initialization to the concrete interrupt request handling. The model allows random generation of different tasks for initialization and list of action, whose execution (respectively ordering) allows controlling the learner's knowledge.

In the future the model could be modified by adding a new window, where the learner, after ordering in a correct sequence the list of actions for interrupt request handling, will have to simulate the corresponding sequence of necessary control signals for performing these actions.

REFERENCES

[1] Vasileva,A., A.Smrikarov, T.Hristov. A Conceptual Model of a Virtual Laboratory on "Computer Organization". Proceedings of the ***CompSysTech'2002***, Sofia, 20-21 June 2002.

[2] Левентал,Л. Въведение в микропроцесорите – апаратно и програмно осигуряване, програмиране. Техника, София, 1982.

[3] Stallings,W., Computer Organization & Architecture – Designing for performance, Pearson Education, 2003.

[4] IA-32 Architecture Software Developer's Manual. Intel Corporation, 2001.

[5] Смрикаров,А., Ц.Василев, И.Цанков, С.Смрикарова. Едночипови микрокомпютри, Авангард Принт, Русе, 2000.

ABOUT THE AUTHORS

Anelia Vasileva, MSc, Department of Computer Systems and Technologies, University of Rousse, Phone: +359 82 888 276, E-mail: ASVasileva@ecs.ru.acad.bg.

Assoc.Prof. Angel Smrikarov, PhD, Department of Computer Systems and Technologies, University of Rousse, Phone: +359 82 888 249, E-mail: ASmrikarov@ecs.ru.acad.bg.