Software Aspects of Intel Pentium Simulator Development

Elena I. Zaharieva-Stoyanova[1], Radoslav At. Atanasov[2], Erkan Sh. Shakir[3]

Abstract **- The simulation technique is often used in computer architecture development. The software simulators could be used as a tool for studying the architectures' models and functionality. In this paper a software simulator named PipeSimul is represented. It shows the Intel Pentium processors' structure and functionality. This paper treats the software problems related with simulator development. The simulation of instructions execution is realized by independent module, represented in this paper. The Graphics User Interface is represented, too.**

Key words **- simulators, computer architecture, Intel Pentium processor, pipelining**.

I. INTRODUCTION

The simulation technique is often used in computer architecture development. The software simulators could be used as a tool for studying the architectures' models and functionality.

The software simulation method is used because of the following reasons:

- Through the circumstances it is the one way of observing of insupportable processes
- Software simulation is cheaper; its realization is faster than hardware execution.
- It has more flexibility
- Software simulation allows to investigate and to modify the simulating object through its design.
- The simulator is a tool for a dynamic optimization of the models, who are not able to be analysed by theirs mathematic description.
- The simulation allows viewing processes hidden into big and complex modules.
- The simulation technique allows realizing virtual, not existed models used for an education.

The development of up-to-date processors' architecture is closely related with instruction-level parallelism and superscalar concepts [2]. Studying of these concepts for theirs application in a real architecture needs a software simulation. This paper treats the problem of software simulator development. The simulator represents Intel Pentium architecture.

[1]Elena I. Zaharieva-Stoyanova is with the Department Computer Systems and Technologies, Technical University of Gabrovo, 4 H. Dimitar, 5300 Gabrovo, Bulgaria, E-mail: zaharieva@tugab.bg.

[2]Radoslav At. Atanasov is a student with the Department Computer Systems and Technologies, Technical University of Gabrovo, 4 H. Dimitar, 5300 Gabrovo, Bulgaria

[3]Erkan Sh. Shakir is a student with the Department Computer Systems and Technologies, Technical University of Gabrovo, 4 H. Dimitar, 5300 Gabrovo, Bulgaria

The Intel Processors are among of most used processors in the computer systems. Developing IA-32 architecture, Intel Corporation introduces superscalar technique in the Pentium processor. The Intel Pentium is the first processor with superscalar architecture. The next steps of IA-32 architecture development are coming of P6 and NetBurst processor architectures [1],[3],[7],[8].

In this paper a software simulator named PipeSimul is represented. It shows the Intel Pentium processors' structure and functionality. The objective of new simulator creation is to show the base concepts of the Intel Pentium processors working by means of short assembler programs. This simulator could be used also for the evaluation of source code efficiency. It will be applied in the hirer-school education to show how superscalar architecture works [4],[5],[6].

This paper treats the software problems related with simulator development. The simulation of instructions execution is realized by independent module, represented in this paper. The Graphics User Interface is represented, too.

II. THE PIPELINE EXECUTION IN INTEL PENTIUM PROCESSOR

The Intel Pentium processor is the first Intel processor with a superscalar architecture. It has two execution pipelines to achieve superscalar performance. Two 5-stages pipelines, known as U and V, together can execute two instructions per clock. In comparison with 80486, the on-chip first-level cache was doubled, with 8 KB devoted to code, and another 8 KB devoted to data. Branch prediction with an on-chip branch table was added to increase performance in looping constructs. The Pentium processor also implements 8-stages pipeline for float-point instructions [9].

The U pipeline is known as a main pipeline. It can execute all 80x86 and Pentium instructions. The V pipeline is used just for hardware executed instructions. The pipelines structure in given on fig. 1. The U and V pipelines have 5 stages:

- Instruction Fetch (IF) stage is a common for both pipelines. It fetches two instructions from the first-level (L1) cache. The active buffer attempts to be filling with 16 bytes code. If there is a JUMP or CALL instruction (JMP, Jcc, and CALL), the branch prediction is activated.
- Decode 1 (D1) stage is common for both pipelines, too. D1 stage consists of two parallel decoders. The Pentium processor determines whether two consequent instructions can execute together. If it is possible, the first instruction is loaded into U pipeline and the second instruction is loaded into V pipeline.
- Decode2 (D2) is separate for U and V pipeline. It determines data addresses.

• Execution (EX) stage executes instructions. The instructions run to this stage together but it is not necessary to leave it together. In this case, the instruction in U pipeline has to leave this stage first.

• Write Back (WB) is the last stage, where the results are written into registers and the flags are changed.

The results from the branch prediction could be controlled at the stage EX (for JMP and CALL instructions), or at the stage WB (for conditional branches).

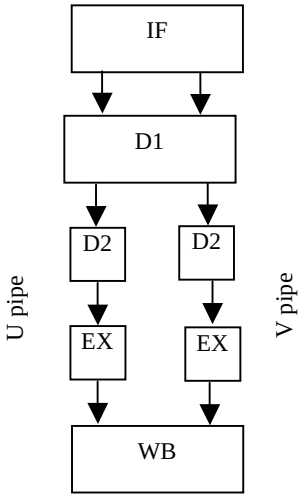

Fig. 1 The structure of pipelines

III. Base structure of Intel Pentium simulator

PipeSimul is a program simulating the superscalar architecture of Pentium processors. Its purpose is to visually show the work of the Pentium's pipes and maintain execution of assembly sources step by step, showing the state of registers, flags, data and stack segments.

PipeSimul is MDI based application. Using MDI architecture the application might be structured according to the information for the different needs of the user. The information is placed in the child windows of the program. These child windows are based on separate View classes and Document classes. This provides additional flexibility in resizing, rearranging, closing and showing windows, which are in relevance. For example, scrolling of code window PipeSimul automatically scrolls the window showing the pipes' structure. Certain Windows can be closed or minimized, when not in use. In this way the application benefits from tiling and cascading. The base structure of PipeSimul is introduced on fig. 2.

PipeSimul is compiled on Microsoft Visual C++. It's based on an open-source project called SoftWire [10].

It is a class library written in object-oriented C++ for compiling assembly code. It can be used in projects to generate x86 machine code at run-time as an alternative to self-modifying code. Its classes are used to parse the assembly source code and to generate the internal structures of the assembler. Then they are used to emulate Pentium's work, especially the pipeling, simulating the execution of the instructions as represented in the internal structures (classes).

PipeSumul is a program demonstrating the work of pipes with its specific features. This application is not a full Pentium simulator. It is not necessary to show all processes in Intel Pentium [6]. In the first version the simulator assumes the following restrictions:

- Simulator executes only real mode applications.
- Simulator uses near program model, so only one code segment and therefore only near jumps and calls. Only one data and stack segment are shown, too.
- Simulator can execute the most used instructions. Only the following instructions: Data transfer: MOV, PUSH, PUSHA, PUSHAD, PUSHF, PUSHFD, POP, POPA, POPAD, POPF, POPFD; Arithmethical instructions: ADD, ADC, SUB, SBB, INC, DEC, CMP, NEG, MUL, IMUL, DIV, IDIV;Logical instructions: AND, OR, XOR, NOT, TEST; Shift and rotation: SHL, SAL, SHR, SAR, ROR, ROL, RCR, RCL, SHLD, SHRD;Instruction flow control: JMP, CALL, RET, Jcc, LOOP, LOOPZ, LOOPNZ; Flag manipulation: STC, CLC, CMC, STD, CLD, STI, CLI, SAHF, LAHF.

These instructions are included in Reduced Instruction Set Computer (RISC). Pipelining is a feature of RISC architecture. Therefore, there is no reason to simulate complex instructions execution.

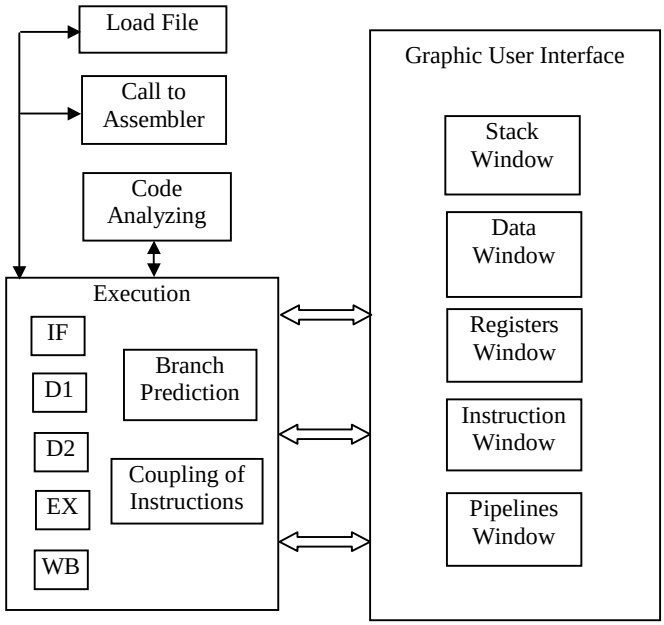

Fig. 2 The base structure of the simulator

IV. Simulation of instruction execution in Intel Pentium

Instruction execution simulation is realized by software module named PS. This module is a part of PipeSimul application. PS is realized as C++ source files closed in namespace Simulator. It this manner the integration of PS module in another software module is easer because the conflicts between identifiers are avoided.

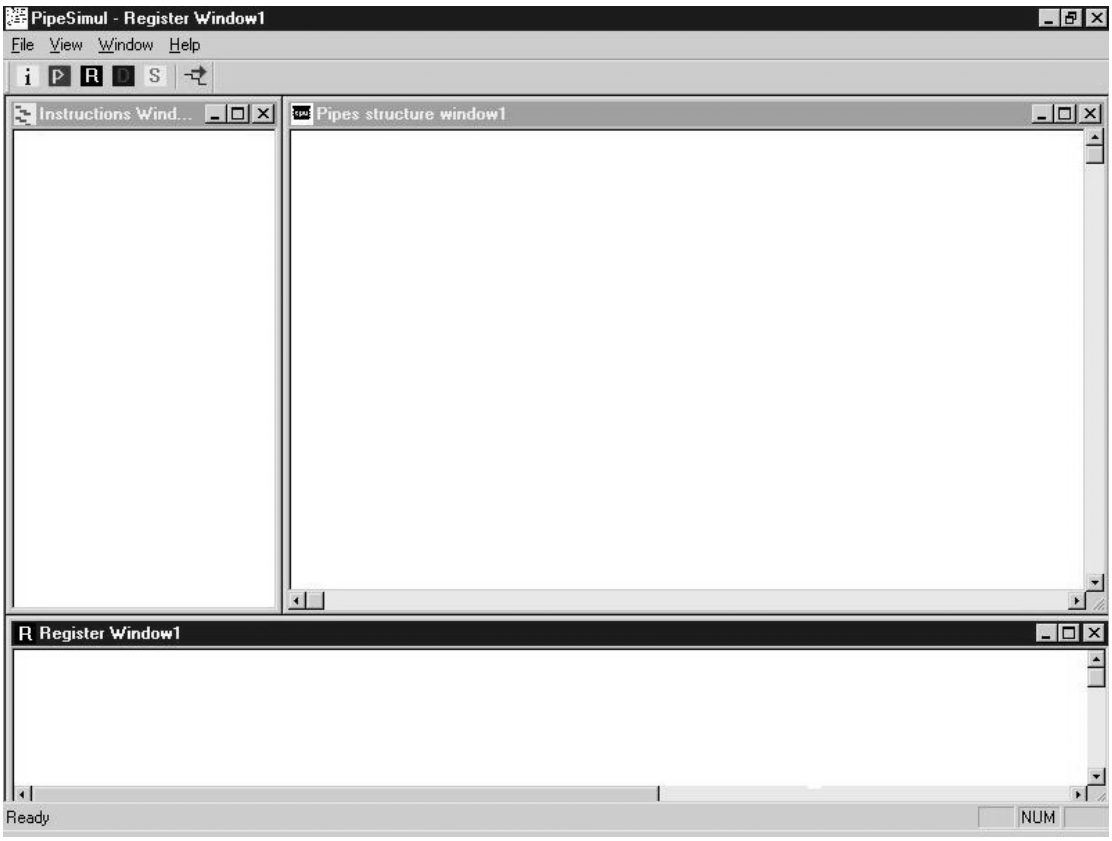

Fig. 3. The PipeSimul application window

The PS module is designed as independent on PipeSimul application. Thus, its integration to another software simulator or to the other similar application is possible.

To use the PS module, it is necessary to instant the object of class Simulator::TSimulator. The constructor finds the path the base module directory.

Simulator::TSimulator object creates an object of class Simulator::TProcessor. Class TProcessor has an array of pointers to class Simulator::TPipe. Class TPipe represents pipelines. In this case, the pipeline numbers is two. When the code has to know this number, it might to bring the value of the constant named Simulator::TProcessor::PIPES_COUNIt is possible to change the source code for more than two pipelines. In this stage of the application development, the most parts of the code rely that the number of pipelines is two.

The instances of class Simulator:TRAM and class Simulator::TIOSpace are created additionally. They serve like as a simulation memory and an I/O address space.

They inherit the base class Simulator::TAbstractDataSpace.

The method Simulator::TSimulation::LoadFile() has as parameter the path to a source file consisted assembler file. The method loads file to memory and prepares it for simulation. It includes: call to Assembler, code analyzing, and program loading to memory as binary code.

The method Simulator::TSimulator::Clock() simulates one cycle. It is all needed for an instruction stream control by the software application.

The pipelines software realization is divided between classes Simulator::TProcessor and Simulator::TPipe. It includes five virtual methods: DoIF, DoD1, DoD2, DoEX, DoWB. The first two methods belong to class Simulator::TProcessor, the other three methods belong to class Simulator::TPipe. These functions correspond to the fifth pipeline stages in Intel Pentium.

This development stage of the simulator executes the instructions simultaneously. The development provides to simulate U and V-pipeline functionality by using classes Simulator::TUPipe and Simulator::TVPipe.

IF stage working represented by DoIF method is to fill 16-bytes buffer memory.

D1 stage realized by DoD1 creates an object of TInstructionSet class. This class loads the instruction codes. At D1 stage, the object of Simulator::TDecoding class is created to decode the instruction according to the Simulator::TInstruction object definition.

This approach saves the creation of an extra source code, it also allows dynamic changing of instruction set.

The method DoD2 computes the addresses and extracts the operands for an instruction execution.

Methods DoEX and DoWB are realized particularly at this development stage.

V. Simulator Graphics User Interface

The Graphics User Interface of the simulator as software application is given on fig 3. The most used windows are given:

- Instruction Window - it contains the executed program code

- Pipelines Window - it represents instruction execution through pipes.
- Register Window - it shows the registers' content.

There is possibility to open Data segment Window and Stack segment Window.

The names of created classes and theirs base classes are given at the table 1.

TABLE 1

Class	Base Class	File
DataSegment	CView	DataSegment.h DataSegment.cpp
DataSegmentDoc	CDocument	DataSegmentDoc.h DataSegmentDoc.cpp
InstrFrame	CMDIChild Wnd	InstrFrame.h InstrFrame.cpp
Instruction	CView	Instruction.h Instruction.cpp
InstructionDoc	CDocument	InstructionDoc.h InstructionDoc.cpp
RegFrame	CMDIChild Wnd	RegFrame.h RegFrame.cpp
Register	CScrollView	Register.h Register.cpp
RegisterDoc	CDocument	RegisterDoc.h RegisterDoc.cpp
Stack	CView	Stack.h Stack.cpp
StackDoc	CDocument	StackDoc.h StackDoc.cpp

VI. CONCLUSION

Software simulation is often used technique in computer architecture development. This paper represents Intel Pentium simulator development. The key points of simulator creation as software tool are: Graphics User Interface development; instruction execution simulation; simulation of branch prediction logic.

This paper represents the solution of instruction execution simulation and Graphics User Interface creation.

The execution of assembler instructions is simulated by module named PS. It uses an open-source object called SotfWire for an assembler code compiling. The PS is designed as independent module. It may be used in another software simulator or similar application.

REFERENCES

[1] Hinton G., D. Sager, M. Upton, D. Boggs, D.Carmean , A. Kyker, P. Roussel, The Micro architecture of the Pentium 4 Processor, *Intel Technology Journal Q*, 2001.

[2] Hlavicka J, Computer Architecture, CVUT Publishing house, 1999.

[3] Keshava J., Vl. Pentkovski, Pentium III Processor Implementation Tradeoffs, *Intel Corp., Intel Technology Journal Q2*, 1999.

[4] Zaharieva-Stoyanova E., Simulation Models Of Pipelining in Intel Pentium Processors, IEEE-TTTC International Conference on Automation, Quality and Testing, Robotics, Cluj-Napoca, Romania, 2002, pp. 373-378.

[5] Zaharieva-Stoyanova E., Simulation of Pipelined Data Processing in Intel Pentium Processor, CompSysTech, Sofia, 2002, pp I.14-1 –I.14-5.

[6] Zaharieva-Stoyanova E., R. Atanasov, E. Shakir, V. Frunze, Software simulator of Intel Pentium Architecture, Advanced Control Theory and Applications, June 16 – 29, Plovdiv – Gabrovo, Bulgaria, 2003, pp. 91-95

[7] A Detailed Look inside the Intel NetBurst Micro-Architecture of the Intel Pentium 4 Processor, Intel Corporation, 2000.

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

[9] Pentium, NiSoft Ltd ,1998.

[10] Capens N., http://softwire.sourceforge.net.