A&QT-R 2004 (THETA 14)
2004 IEEE-TTTC - International Conference on Automation, Quality and Testing, Robotics
May 13 – 15, 2004, Cluj-Napoca, Romania

# SOFTWARE SIMULATION OF BRANCH PREDICTION IN INTEL PENTIUM PROCESSOR SIMULATOR

**Elena Zaharieva-Stoyanova**

Technical University of Gabrovo, Bulgaria
4 H.Dimitar str., 5300 Gabrovo, Bulgaria,
phone +359 66 223 529, e-mail Zaharieva@tugab.bg

**Abstract:** The simulation is often used technique in computer architecture development. The design of processor simulators as software tools requires a usage of different software techniques to simulate and to visualize the hardware processes in computer architectures. This paper represents the software aspects of the problem**: realization of branch prediction logic functionality**. The suggested software solutions use some possible schemes and they have an application in software simulator of Intel Pentium processor named PipeSimul. The base structure and functionality of this software application are represented, too. The suggested schemes could be included also in other software simulators of processors' architectures.

**Key words:** branch prediction**,** superscalar architecture, IA-32 architecture, Intel Pentium processor, and simulators.

## 1. INTRODUCTION

The simulation is often used technique in computer architecture development. The software simulators are tools for studying of the architectures because of the possibility to visualize theirs functionality. The simulators could be used also for optimizations algorithms and for estimation of the different architecture solutions [2].

Development of processor simulators as software tools needs solutions of different tasks. It is necessary to use different software techniques to simulate and to visualize the hardware processes in computer architecture. To show how does pipeline work, for example, the developer has to use visual software tools.

The simulators need also modules to imitate techniques as: decoupling of instructions, speculative executions, and branch prediction.

This paper represents the software aspects of the problem: **realization of branch prediction logic functionality**. The suggested software solutions use the possible schemes and they have an application in software simulator of Intel Pentium processor. Although, the offered decisions consider with the data structure and a functionality of particular software application, they could be included also in other software simulators of processors' architectures.

## 2. STRUCTURE AND FUNCTIONALITY OF INTEL PENTIUM SIMULATOR

The Intel Pentium processor is the first Intel processor with a superscalar architecture. The structure and functionality of Intel Pentium architecture software simulator is represented in [3],[4]. As a software application the simulator is named PipeSimul. This paper treats the software problems related with creation of a processor

simulator as software application. Because the represented solution refers to particular problem, the simulator will be represented briefly.

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. MDI architecture displays a lot of information and this information might be structured in the program 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 structure of the simulator is shown on fig. 1.

The Graphics User Interface (GUI) consists of the following forms:

- ***Instruction window*** - holds a list of instructions which user can load from file and to see how processor executes them

- ***Pipelines window*** - shows graphically how processor executes instructions, how it passes through pipeline and how many clocks takes each one of them. Every phase is represented by different color so this allows the user to trace the execution process in convenient way.

- ***Registers widow*** – it shows the state of following registers: EAX, EBX, ECX, EDX, ESP, EBP, ESI and IP. It also displays the state of most used flags.

- ***Data window*** – in the initial version of PipeSimul it contains hexadecimal dump of data segment contents. It might be improved in more readable view, but this is left for the future versions.

- ***Stack window*** – as Data segment window it shows hexadecimal dump of the stack segment contents. On start-up it is closed but the user can show it from the View menu.

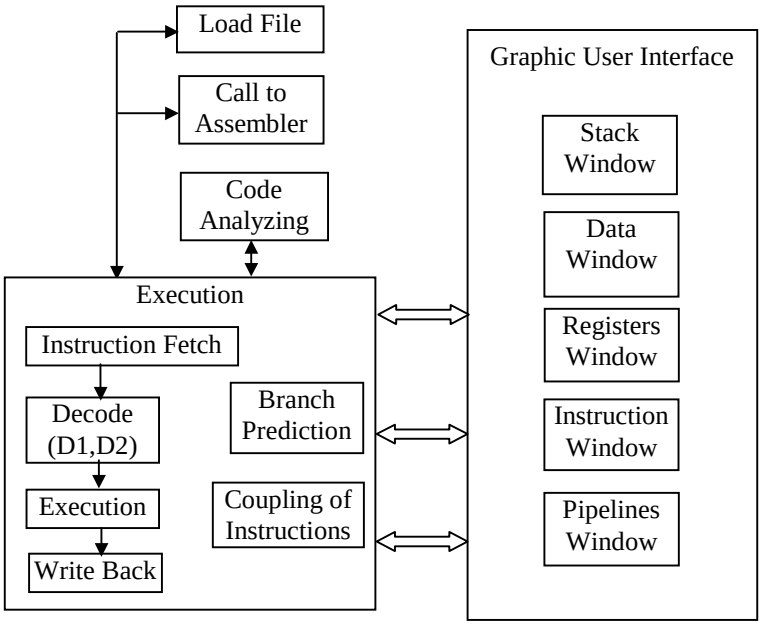


Fig. 1. The base structure of the simulator

The instruction execution is realized by independent module. The module has be extended with a module simulating branch prediction logic functionality.

### 3. BRANCH PREDICTION FUNCTIONALITY

Branch prediction is an advanced computing technique increasing the performance by keeping the execution pipelines full. This technique reduces the influence of control hazard in execution pipeline. There are two base methods***: Static Branch Prediction*** and ***Dynamic Branch Prediction***. The second one is most useful in the modern processors architecture [1], [2].

Most of the dynamic branch prediction methods use some sort of cache to generate a guess: "is the branch will be taken or not" based on past execution of the branch.

A branch prediction buffer is a 2-bits wide memory indexed by lower bits of the Program Counter. Each entry indicates whether the last executed branch was taken or not. If the prediction is correct, there will be no branch penalty. If the penalty occurs the bits have to be undated.

The state diagram of the two bits branch prediction mechanism is shown on fig.2. It is obvious that prediction will differ from the previous one after two misspredictions.

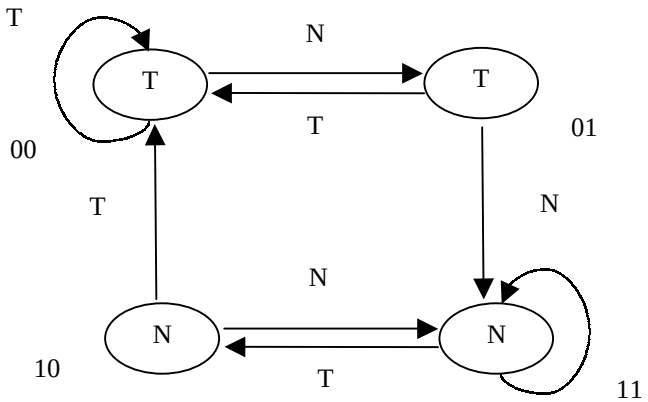


Fig. 2. Branch prediction scheme

Another possibility is using of Branch target Buffer (BTB). BTB is a small cache memory indexed by current state of Program Counter and used to predict the next instruction address in the case of cache hit. On a cache miss, an instruction and a next address are placed only if the instruction is a branch.

Intel Pentium processor supports branch prediction using Branch Target Buffer (BTB). The branch prediction logic starts when branch instruction (JMP, Jcc, CALL, RET) comes to the first stage of the pipeline. This is an IF (Instruction Fetch) stage. The results from the branch prediction could be controlled at the 4th stage EX (Execite) for JMP and CALL instructions, or at the 5th stage WB (Write Back) for conditional branches.[9]

To efficiently predict branches, Intel Pentium uses two prefetch buffers. The first buffer prefetches code in a linear fashion - the next execution step, while the other prefetch instruction based on the address in BTB. As a result, the needed code is always prefetched before it is required for execution.

The branch prediction algorithm uses a two-bits scheme. The logic goes through following states, as shown on the diagram:

***highly non-taken branch -> lowly non-taken branch -> lowly taken branch -> highly taken branch***

The state diagram on fig. 3 shows the relation between bits value and a branch status. Initially, the branch prediction logic considers that the branch the branch is highly non-taken. It means that the bits value is 00. After two sequential misspredictions the branch status will be changed.

The prediction algorithm is able to foresee simple branches. It also supports more complex branch prediction. This is accomplished by storing multiple branch address in BTB. Branch Target Buffer has 256 entries. It is design to store 256 branch addresses [6],[7].

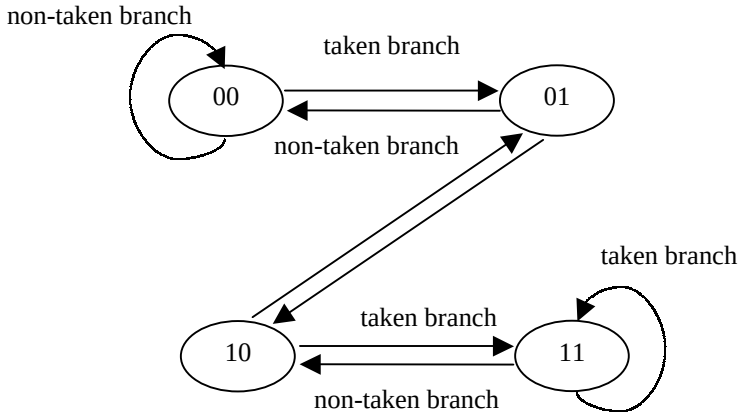


Fig. 3. Branch prediction scheme in Intel Pentium

## 4. SOFTWARE SIMULATIONS OF BRANCH PREDICTION LOGIC

One of the problems related with a creation of Intel Pentium simulator is how to represent a functionality of branch prediction logic. Software realization is closely related with an instruction describing.

The executed by simulator instructions are placed into a code buffer. They are Instruction class objects. This class has members describing: ***instruction opcode***, ***first operand***, ***second operand***, and ***instruction type***.

### *4.1. Software simulation of branch prediction logic by using data structure*

One possible solution of branch prediction problem is adding a new member – two-bits field, who stores an information for taken branch or non-taken branch. In this case, each instruction will has such member, although the instruction might not be a branch.

The instruction execution algorithm checks the type of the current instruction, first. If the instruction is JMP, Jcc, CALL, or RET, the algorithm check also bit-field for taken or non-taken branch. According to the result, the next instruction address will be extracted. The status of the branch will be displayed, too.

This solution is possible in software simulator but it has two disadvantages:

- Objects of Instruction class occupy redundant memory because not each instruction is a branch.
- This scheme not simulates the branch prediction logic working exactly.

This scheme is proper only for a simulator and may be used to facilitate programmers. It eliminate necessarily of the Branch Prediction module.

### *4.2. Software simulation of Branch Prediction Buffer*

Another possible solution is to imitate a Branch Target Buffer. BTB is represented as 256 entry buffer memory. As it is shown on fig. 4, each entry keeps the following information:

- address of branch instruction;
- two-bits field to indicate taken or non-taken branch;
- address of the next instruction.

| branch address | taken branch | next instruction address |
|---|---|---|

Fig. 4. Branch Target Buffer entry

The address of instructions is represented as on offset in the instruction code buffer. If the current instruction is a branch, its address might be found in BTB. On address miss, it has to be entered into the buffer.

This solution does not occupy redundant memory but it needs an address-searching algorithm.

The Branch Target Buffer in simulator might be addressed by the offset of the current instruction in Code buffer. The BTB size has to be enough for the all instruction in Code Buffer. The BTB entry has the following structure:

- two-bit field for taken or non taken branch;
- offset of the branch target in Code Buffer.

This scheme use redundant memory but it is near to branch prediction logic functionality. The address of branch instruction could be found easy in BTB buffer.

The Branch prediction logic work by following algorithm:

1. *If the current instruction is a branch, check the offset in BTB.*
2. *If the most significant bit is equal to 1, bring from BTB the next instruction displacement. Otherwise, the next instruction address will be kept.*
3. *On missprediction, stall the pipelines, and start with a correct address.*
4. *Update the BTB status according to scheme on fig. 3.*

This algorithm is an imitation of branch prediction logic working. It is possible to realize it in software simulator. Moreover, it is near to real branch prediction functionality.

## 5. CONCLUSION

This paper treats the problem of software development of processors' simulators. The subject is the presentation of branch prediction logic functionality as a part from a software application. Some possible solutions of software imitation of Branch Prediction are given. They have an application in software simulator of Intel Pentium processor named PipeSimul.

The structure of a software simulator PipeSimul also is represented in this paper. 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, and data and

stack segments. This kind of simulator would be very useful in hirer-school education to illustrate the pipelining in a superscalar architecture. Moreover, it shows a real existing often-used processor as Intel Pentium.

The suggested software solutions use the some possible schemes and they have application in software simulator of Intel Pentium processor. They could be included also in other software simulators of processors' architectures.

## 6. REFERENCES:

[1] Hlavicka J., (1999), Computer Architecture, CVUT Publishing house.

[2] Parikh D, K. Skadron, Yan Zhang, M. Barcella, M. R. Stan, Power Issues Related to Branch Prediction, http://lava.cs.virginia.edu/

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

[4] Zaharieva-Stoianova, E., (2002), Simulation of Pipelined Data Processing in Intel Pentium Processor, International Conference CompSysTech, 20-21 Jun, 2002, Sofia, pp. I.14-1 – I.14-5.

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

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

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

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

[9] Pentium, (1998), NiSoft Ltd.