Performance Evaluation of Superscalar Processor Architecture Through UML

Taskeen Zaidi[1] and **Vipin Saxena**[2]

Submitted in May 2012, Accepted in October 2012

Abstract - In the current scenario, most of the applications are based upon graphical user interface and dependent upon the object-oriented technology. Software Industries are interested to convert old structured based softwares into object-oriented based softwares and also to reduce the lines of the code of application for reduction in the execution time of application. Therefore, it is a big challenge to reduce the execution time of the application based upon the object-oriented technology. The present work deals with the reduction of execution time for the superscalar machine by the use of object-oriented approach. A well known modeling language i.e. Unified Modeling Language (UML) is used to model the superscalar pipeline architecture. UML class and sequence models are designed before computations of the execution time and computed results are depicted in the form of tables and graphs. The comparisons are also made by taking the two object-oriented programming languages.

Index Terms - Superscalar pipeline architecture, performance evaluation, class model, sequence model and unified modeling language.

1. INTRODUCTION

Pipelining is one of the important techniques which have been implemented to improve the performance of a processor. It allows the concurrent execution of several instructions. A task or program or process is divided into sequence of subtasks and each task is executed by a specialized hardware stage which operates concurrently with other stage in pipeline. There are several categories of pipeline like arithmetic pipeline, instruction pipeline, memory access pipeline and superscalar pipeline. Superscalar pipeline architecture can start two or more instructions in parallel in one core, and independent instructions may get executed out-of-order. For parallelism, scalabilityand programmability, [1] is an important release which describes these aspects with increasing system resources and accordingly to parallel, vector and scalar instructions. Mano [2] describes the computer organization and design as well as programming using basic components.

Patterson and Hennessey [3] covers the most fundamental areas of computer architecture including recent technologies, like multicores and multiprocessors.

The depth treatment with the implemented details of pipelined processors and memory systems; the "micro architecture" of the modern computers and microprocessors by exploring the techniques for solving design problems inherent in computers with high level concurrency as the demand for a memory system with low latency and high bandwidth are described by Cragon and Saini [4,5].

Unified Modeling Language (UML) is a general purpose modeling language which is used to model various kinds of the research problem widely accepted by the software professionals and created by Object Management Group (OMG [6,7] and development stages are well explained by Booch et al. [8]. The fundamentals of UML using hands-on projects, drills and mastery checks which illustrates how to read, draw, and use this visual modeling language to create clear and effective blueprints for software development projects are explained by Roff [9]. UML is also used to model the concurrent distributed and real time applications which help the researchers to leverage the powerful flexibility and reliability of the system. UML also helps the designers at every stage of the analysis and design process and offers exceptional insight into dynamic modeling, concurrency and distributed applications designing and performance analysis of real time designs [10]. By using distributed computing, the performance of processors for different object-oriented software system framework has been measured by Saxena et al. [11]. They have chosen two types of object-oriented software system frameworks C# based on Microsoft.NET framework and Visual C++ based on Microsoft Foundation Classes (MFC) and computed the performance of these two object-oriented languages. The UML modeling for instruction pipeline design by two techniques i.e. data forwarding and without data forwarding are explained by Saxena and Raj [12]. The modeling and specification of floating point numbers are implemented by Boldo et al. [13]. It extends an existing tool for the verification of C programs, with the new notations specific to the floating point arithmetic. It also provides a way to perform the full formal proof by use of COQ proof assistant and an open framework which is implemented to other floating point models. But the main limitation is that it is applicable only to programs using basic. The IEEE standard is the most widely used standard for floating point and arithmetic representation. It is implemented on most of Central Processing Units (CPU's) and Floating Points Units (FPU's); explained with basic and extended floating point number formats, operations such as add, multiply, divide, square root, etc. It is also used to implement the conversion between integer and floating point formats, but, it does not specify the decimal strings and integers, interpretation of NAN's and conversion of binary to decimal to and from extended format [14]. Saxena and Shrivastava [15] have attempted to increase the performance of arithmetic

[1, 2] Department of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow (U.P.), INDIA
E-mail: [1]taskeenzaidi867@gmail.com and [2]vsax1@rediffmail.com

pipeline especially for floating point computations after designing the complete UML model of static arithmetic pipeline design. They presented UML diagrams to model the system architecture and timing behavior. Saxena and Shrivastava [16] also presented floating point computations by using nonlinear arithmetic pipelining for instruction coupled on Visual C++ and Visual C#. The computations are performed inside a loop by varying the number of repetition of terms for getting their sum. In the current scenario, distributed computing approach is most popular approach and in this regards, a comparative study of the distributed computing paradigms is presented in [17]. Quality of services is one of the major issue for the distributed computing applications and these are described by Mohan et al. [18] for the process centric development. The design patterns for the service oriented architecture implementation are described by Tere and Jhadav[19].

In the present work, UML is used to model class and sequence diagrams for superscalar pipeline architecture which can execute two or more instructions in parallel and authors evaluated the performance of the two object-oriented languages like Visual C++ and Visual C# and some of the important observations are recorded in the form of table and graphs.

2. BACKGROUND

2.1 Process Definition

Let us first explain the process which is considered as a program which is to be executed. It can be defined as a unit of work in modern time sharing systems. For defining the process processing element is needed to be defined as stereotype and is used to handle some modeling elements based on UML base classes. A UML Class for process is shown in figure1 and is identified by its own identification number represented as Process-id. The other attributes are Process-size for the size of a process; Process_in_time and Process_out_ time are for start at out time of the process. The attribute Process_priority controls the priority of the incoming process. These attributes work on the operations like Process_create(), Process_delete, Process_ update, Process_ join, Process_suspend, and Process_synchronize. The visibility modes along attribute and operation are also shown in the figure. A stereotype processing unit is also depicted in figure 2, the instance and multiple instances of class Process class are shown in Figures 3(a) and 3(b), respectively. The process may consist of segments of code whose identification numbers are generated; recorded into a list and granted processing unit as per the priority of that segment code behaving as a process. The segments may be synchronized with the processing unit as per the time of completion of that segment; therefore, multiple instances of a process are shown in figure 3(b).

2.2 Thread

A thread is defined to control a block of code that runs concurrently with other threads within same process. It is a sequential flow of instructions and it is considered as lightweight process. It is easily handled in object-oriented way. Threads run simultaneously in process and can access the same object to implement their functionality.

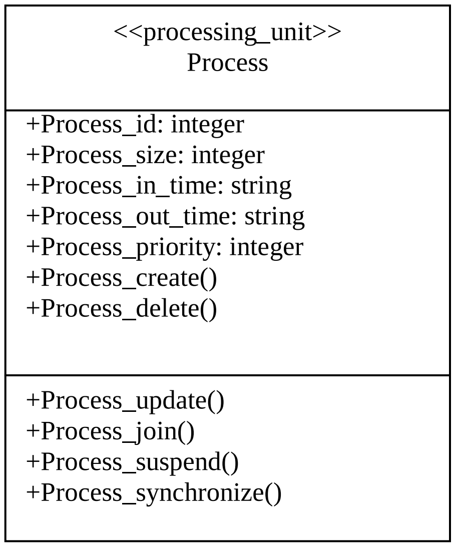


Figure 1: UML class diagram of a process

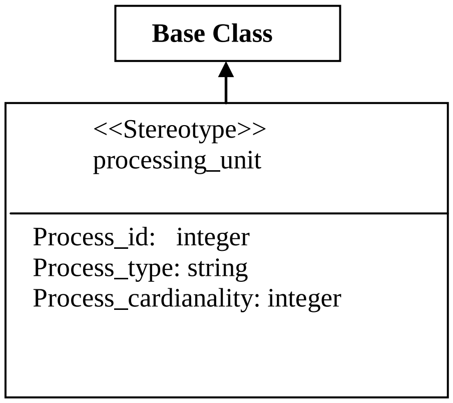
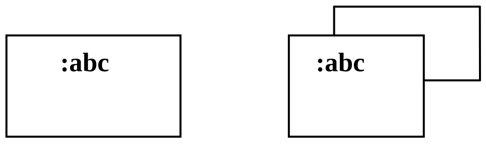


Figure 2: UML class for processing unit



3(a) 3(b)
Figure 3a: Single instance, Figure 3b: Multiple instances

In the current scenario, most of the window based applications are based upon the thread concept as system supports synchronization of sub tasks of a process. Threads are initialized and after the use these are automatically destroyed, therefore, it has a life cycle. Object-oriented representation of thread is shown below in figure 4, in which it is identified by an attribute called as Thread_id. The other attributes associated with thread and thread operations are also shown below in the figure 4.

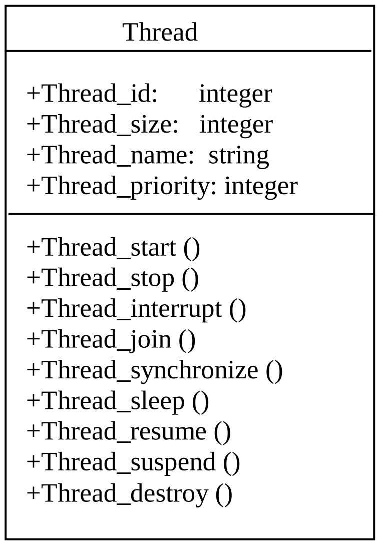


Figure 4: UML Class Diagram of a Thread

2.3 Superscalar Processor Architecture

Superscalar processor architecture has a versatile design with the two pipelines and it can issue the instructions per cycle, if there is no resource conflict and no data dependence problem. Both pipelines have four processing stages namely fetch, decode, execute and store.

Each pipeline has its own fetch, decode, execute and store unit. The two store units can be dynamically used by the two pipelines, depending upon its availability at particular cycle. It has four functional unit adder, multiplier, logic and load unit. These all functional units are shared by pipelines on dynamic basis. There is a lookahead window with its own fetch and decoding logic. Lookahead window is used in case of out of order instruction to achieve better pipeline throughput.

3. UML MODELING FOR SUPER SCALAR PIPELINE DESIGN

3.1 UML Class Diagram

The figure5 shows the architectural model of superscalar processor. The class process interacts directly with PEC which executes the assigned task. The PEC controlled the process by exchanging message between classes processor and memory. The processor class has two cores i.e. Core1 and Core2 and each core has many components which help in process execution as shown in figure.

In this figure, class L2_cache is shared by two cores and caches instruction through the class I Cache whereas D_cache caches the data, which itself is subclass of L1_cache. The class ALU computes integer arithmetic and logical operations; FPU is used for floating point operations as shown in figure. SU is used for storing the outputs. FPU class contains four classes as namely Adder, Multiplier, Logic and Load unit.

3.2 UML Sequence Diagram

The UML Sequence diagram represents the dynamic behavior of system in which objects are interacted with the help of message communications. The vertical line shows the life line of object or dynamic representation of system, a UML sequence diagram is shown in figure 6 for process execution in Superscalar pipeline architecture.

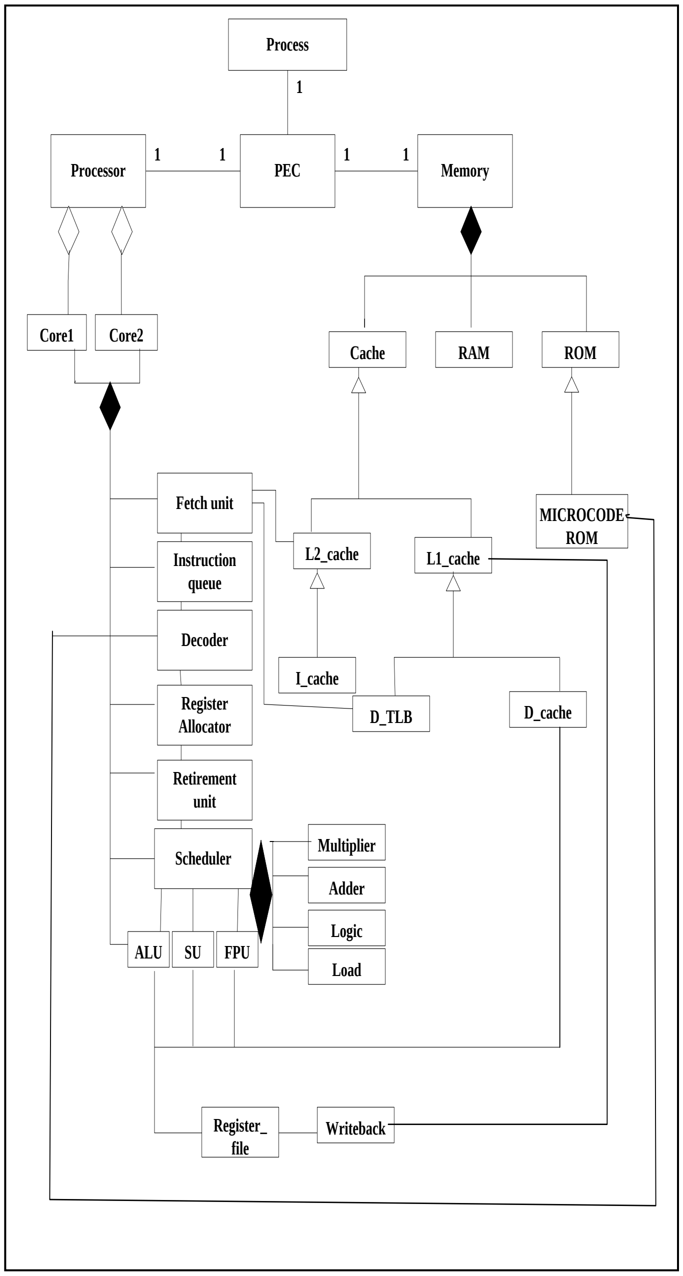


Figure 5: UML class diagram for superscalar process

The processor executes the instructions fastly through execution pipelining, which execute multiple instructions at same time. The instruction fetched, decoded and finally goes to PEC where instructions executed and results store in Registerfile and then Writeback.

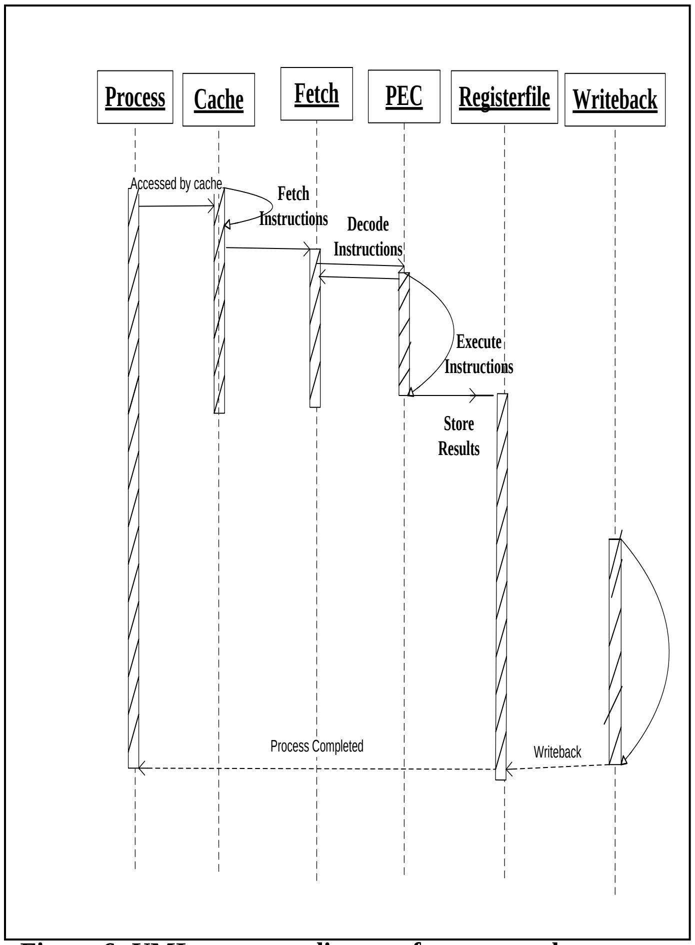


Figure 6: UML sequence diagram for superscalar processor

4. EXPERIMENTAL STUDY

On the basis of above object-oriented design, let us consider the two object oriented languages i.e. VC++ and VC# which work on the .Net platform. For these two programming languages, a relative performance of the superscalar processor is computed. In the computation, let us consider N are the independent instructions which can be executed in parallel through pipeline method and k is taken as the time required to execute instructions through m pipeline simultaneously, then the ideal time required by scalar base machine is

T (1, 1) =k + N-1……………………. (i)

The ideal execution time is computed by

T (m, 1) =k + (N-m)/m…………….. (ii)

The computations for ideal execution time are done by taking lines of code varying from 10^2 to 10^5 and these instructions are considered by increasing the size of loop. Execution time is computed by taking average of five runs and results are depicted in the table1. As expected lines of code are increasing, execution time is also increasing but if one compares VC++ andVC#, for long computations in milliseconds, it is observed that VC++ takes lesser time in computation than VC#. These results are also graphically represented infigures 7 and 8 given on next page for 10^2, 10^3 and 10^4,10^5 lines of code (LOC), respectively.

5. CONCLUSION

From the above work, it is concluded that UML is powerful modeling language accepted by software Professionals and also used to represent hardware architecture problems. For the long computations, software professionals are facing the problems for selection of best object oriented Programming language which works well on any kinds of processor architecture.
Therefore, superscalar processor architecture is modeled by the use of UML classes and experimental results are performed by taking two object-oriented programming language like Visual C++ and Visual C# and concluded that Visual C++ is better in comparison to Visual C# as one is performing the long computations.

REFERENCES

[1]. Hwang, K., Advanced Computer Architecture: Parallelism, Scalability, Programmability, Fourteenth Reprint, Tata McGraw-Hill Edition, ISBN-0-07-053070-X-2007.

[2]. Mano Morris, M., Computer System Architecture, Third Edition, Prentice Hall of India Pvt Ltd. ISBN-978-81-203-0855-8, 2007.

[3]. Patterson, A. David and Hennessy, L. John, Computer Organization and Design: The Hardware/Software Interface, Morgan Kaufmann Publishers Elsevier Inc., 2005.

[4]. Cragon, G. Harvey, Memory Systems and Pipelined Processors, Narosa Publishing House, New Delhi, 1998.

[5]. Saini, A., "Design of the Intel Pentium TM Processor", Intel Corporation, IEEE, and Available: http://ieee xplore.ieee.org/stamp/stamp.jsp?arnumber=00393370 (Accessed on 14th March 2012).

[6]. OMG (2001),"Unified Modeling Language Specification", Available online via http://www.omg.org.

[7]. OMG (2002), "XML Metadata Interchange (XML) Specification", Available online via http://www. omg.org.

[8]. Booch, G., Rambaugh, J., and Jacobson, I., The Unified Modeling Language User Guide, Twelfth Indian Reprint Pearson Education, 2004.

[9]. Roff, T., UML: A Beginner's Guide, Tata McGraw–Hill Edition. Fifth Reprint, 2006.

[10]. Gomaa, H., "Designing Concurrent, Distributed and Real Time Applications with UML", Proceedings of the 23rd International Conference on Software Engineering (ICSE'01), IEEE Computer Society, 2001.

[11]. Saxena, V., Arora, D., and Ahmad, S.; "Object Oriented Distributed Architecture System through UML", IEEE International conference on Advanced in Computer Vision and Information Technology (ACVIT-07), Nov. 28-30, ISBN 978-81-89866-74-7, pp.305-310,2007.

[12]. Saxena, V. and Raj, D., “UML Modeling for Instruction pipeline Design”, World Conference on Science, Eng-ineering and Technology (WCSET,2008), www .waset. org/ pwaset (Acessed on 16 NOV,2011).

[13]. Boldo, S. and Filliatre, J.C., “Formal Verification of Floating Point Programs”, 8th IEEE Symposium on Computer Ari-thmetic (ARITH ’07), pp.187-194 . Availabale: http :/ / www. computer.org (Accessed on 16 Nov, 2011).

[14]. Lopez, G., Taufer, M., and Teller, P.J., “Evaluation of IEEE 754 Floating-Point Arithmetic Compliance across a wide range of Heterogeneous Computers”, Proceedings of the 2007 Richard Tapia Celebration of Diversity in Computing Conference, October 2007 , Orlando, Flor-ida ,USA. Available:http://gcl.cis.udel.edu/publication/ conferences/ 007tapia_mlopez.pdf (Accessed on 16 Nov, 2011).

[15]. Saxena, V. and Shrivastava, M., “UML Modeling of Static Arithmetic Pipeline Design”, The ICFAI University Press Vol. 7(1), pp.22-31, February 2009.

[16]. Saxena, V. and Shrivastava, M., “Performance Evaluation of Non-Linear Pipeline through UML”, International Journal of Computer and Electrical Engineering,Vol.2, No.5, pp.860-866, October, 2010.

[17]. Kumar, H. and Verma, A.K., “Comparative study of Distributed Computing Paradigms”, BIJIT – BVICAM’s International Journal of Information Technology, Vol. 1(2), Dec. 2009.

[18]. Mohan, K.K., Srividya,A., Verma, A.K. and Gedela, R.K., “Process Centric development to Improve Qos in Building Distributed Applications”, BIJIT – BVICAM’s International Journal of Information Technology, Vol. 1(1), July, 2009.

[19]. Tere, G.M. and Jhadav, B.T., “Design Patterns for successful Service Oriented Architecture Implementation”, BIJIT – BVICAM’s International Journal of Information Technology, Vol. 2(2), Dec. 2010.

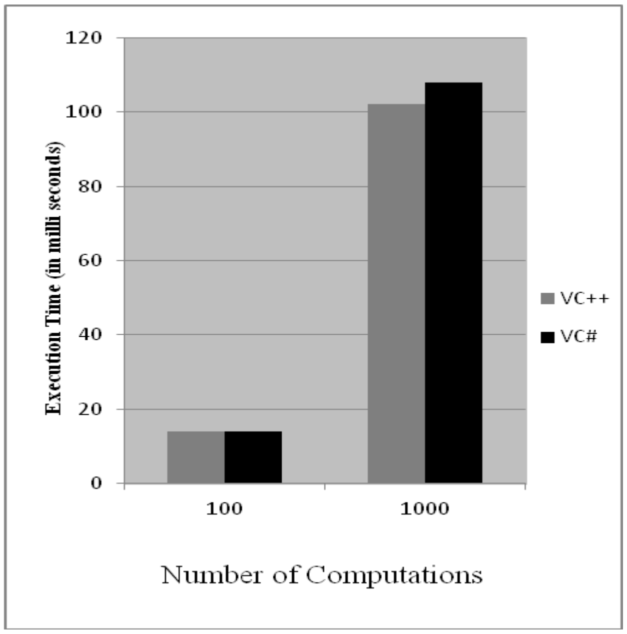


Figure 7: Comparisons for 10^2 and 10^3 Lines of Code

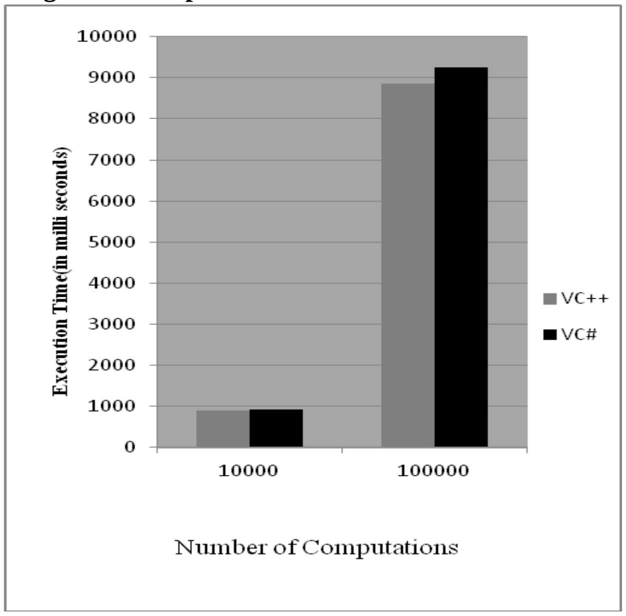


Figure 8: Comparisons for 10^3 and 10^5 Lines of Code

	VC++				**VC#**			
Lines of Code	10^2	10^3	10^4	10^5	10^2	10^3	10^4	10^5
Execution Time (in Milli Seconds)	14.05	92.005	889.0005	8952.00005	14.05	108.005	920.0005	9233.00005
	14.05	109.005	889.0005	8967.00005	14.05	108.005	936.0005	9170.00005
	14.05	108.005	874.0005	8780.00005	14.05	108.005	920.0005	9264.00005
	14.05	93.005	890.0005	8796.00005	14.05	108.005	920.0005	9249.00005
	14.05	108.005	905.0005	8812.00005	14.05	108.005	936.0005	9280.00005
Average Execution Time	14.05	102.005	889.4005	8861.40005	14.05	108.005	926.40005	9239.20005

Table 1: Ideal execution time for superscalar processor