Analytical Models of Instruction Fetch on Superscalar Processors

Sun Mo Kim Jin Ha Jung Sang Bang Choi
Dept. of Electronic Engineering, Inha University
253 Younghyun-Dong, Nam-Gu, Inchon, 402-751, Korea
Tel: +82-32-860-7417, Fax: +82-32-868-3654
E-mail: sangbang@inha.ac.kr

Abstract: This research presents an analytical model to predict the instruction fetch rate on superscalar processors. The proposed model is also able to analyze the performance relationship between cache miss and branch prediction miss. The proposed model takes into account various kind of architectural parameters such as branch instruction probability, cache miss rate, branch prediction miss rate, and etc.. To prove the correctness of the proposed model, we performed extensive simulations and compared the results with those of the analytical models. Simulation results showed that the proposed model can estimate the instruction fetch rate accurately within 10% error in most cases. The model is also able to show the effects of the cache miss and branch prediction miss on the performance of instruction fetch rate, which can provide an valuable information in designing a balanced system.

1. Introduction

The goal of a superscalar microprocessor is to execute multiple instructions per cycle. It relies on instruction level parallelism (ILP) to achieve this goal. However, all of this potential parallelism will never be utilized if the instructions are not delivered for decoding stage and execution stage at a sufficient rate. Moreover, as the performance gap between processor and main memory continues to widen, cache performance is one of the most critical factors affecting the whole system performance.

Enormous amount of research focuses on improving the cache behavior until now. But simulation based methods can not precisely explain the obtained results. Moreover, when a new processor is designed, huge simulations must be performed again with several different parameters. To compensate for these drawbacks, some recent studies showed alternative methods that analytically explained the behavior of cache process. Wallece and Bagherzadeh proposed analytical instruction fetch model and introduced a new fetching mechanism called a dual branch target buffer[1]. But he did not consider the effects of cache miss and branch prediction miss on fetch processing. In fetch stage, if cache miss occurs, cache memory cannot offer instruction blocks any more and must wait for the transfer of another blocks from secondary submemory. Fetch stage must be paused during these cycles. Also, to reduce branch penalties, most recent processors use various techniques such as branch-prediction buffer, branch -target buffer, delayed branch, loop unrolling, etc.. But no method can perfectly resolve branch penalties. If a branch prediction is incorrect, all successive instructions fetched from the wrong path after the branch must be removed from all stages. Therefore, fetching of instructions is constrained by above two major factors: cache miss, and branch prediction miss.

This paper suggests an analytical models that can predict instruction fetch rate for superscalar processors considering cache miss and branch prediction miss. And the proposed model takes into account such factors as branch instruction frequency, cache miss rate, cache miss penalty, branch prediction miss rate, and branch prediction miss penalty. Thus the proposed model can disclose the effects of various factors on instruction fetch. We assumed four cache models which can be applied to superscalar processor. Firstly, in each model, we showed analytical models of instruction fetch assuming no cache miss and perfect branch prediction. Secondly, according to instruction access patterns and occurrences of cache miss and branch prediction miss, instruction fetch processing are divided into various cases. And considering additionally wasted clock cycles, instruction fetch rates of first step are modified.

To prove the correctness of the model, we performed extensive simulations and compared the results with the analytical model. Simulation results showed that the proposed model can estimate the expected instruction fetch rate accurately within 10% difference in most cases. This paper also shows that the decrease of instruction fetch rate is more affected by increase of cache miss rate rather than that of branch prediction miss rate. Similarly, the effects of cache miss penalties are bigger than the those of branch prediction miss penalties. Thus we can know that cache miss must be minimized in designing new superscalar system.

The proposed models are able to show the effects of cache miss and branch prediction miss on instruction fetch rate, and also able to more accurately analyze the relation between cache miss and branch prediction miss.

2. Analytical Models of Instruction Fetch

2.1 Instruction Fetch on Superscalar Processor

Fig. 1 shows the instruction fetching block diagram on superscalar processors. The transfer of instructions from cache memory to pipeline's decoding stage is called *instruction fetch* and the number of instructions fetched per clock cycle is called *instruction fetch rate*. The instruction fetcher is respon-

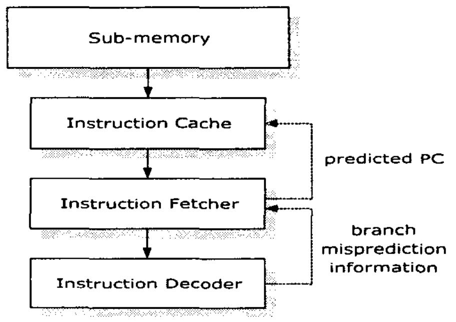

Fig. 1. Instruction Fetching Block Diagram.

sible for determining the new starting PC and sending it to the instruction cache. Instructions after the first branch instruction are invalidated.

Let n be the width of a cache block and b the probability that an instruction is branch instruction. The expected number of instructions which can be sequentially fetched is as follows.

$$L(n,\ b) = n(1-b)^n + \sum_{i=1}^{n} i(1-b)^{i-1}b$$
$$= \frac{1-(1-b)^n}{b} \qquad (1)$$

2.2 Simple Cache

Simple cache has the line size equal to the width of the fetch block. As with all fetching techniques, instruction fetcher can search one cache line each cycle but if there is a branch instruction, instructions after it are invalidated. Fig. 2 shows an example for the simple fetching mechanism.

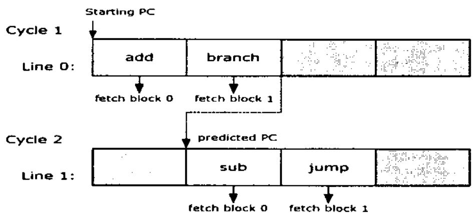

Fig. 2. Simple Fetching Example.

Let $S_i(n,\ b)$ be the probability the starting address in the block is at position i, and $C_i(n,\ b)$ be the probability a branch occurs at position i.

$$S_1(n,\ b) = \{1-C(n,\ b)\} + \frac{C(n,\ b)}{n}$$
$$= 1 - \frac{n-1}{n}C(n,\ b)$$
$$S_i(n,\ b) = \frac{C(n,\ b)}{n},\ 2 \le i \le n$$
$$c_i(n,\ b) = \sum_{j=1}^{i} b(1-b)^{i-j} \cdot S_j(n,\ b) \qquad (2)$$

where $C(n,\ b)$ is the probability that a branch occurs in a cache block.

$$C(n,\ b) = \sum_{i=1}^{n} c_i(n,\ b) = \frac{n}{\frac{1}{b}+n-1} \qquad (3)$$

The expected fetch rate for simple cache is as follows.

$$F_{simple}(n,\ b) = \sum_{i=1}^{n} S_i(n,\ b) \cdot L(n-i+1,\ b)$$
$$= \frac{n}{1+b(n-1)} \qquad (4)$$

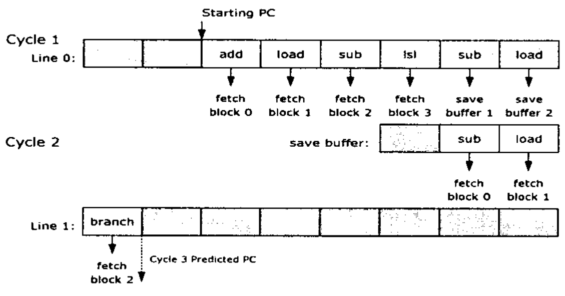

Fig 3. Extended Fetching Example.

2.3 Extended Cache

Fig. 3 shows extended fetching example. This technique extends the instruction cache line size beyond the width of the fetch block and has buffers to save the last n-1 instructions. Next cycle, the instruction fetcher can combine them with instructions that are read from another cache line. The expected fetch rate is as follows.

$$F_{extended}(n, b, m) = \frac{m-n}{m}L(n, b) + \frac{n}{m}F_{simple}(n, b)$$
$$= \frac{(m-n)}{m}\frac{(1-(1-b)^n)}{b}$$
$$+ \frac{n}{m}\frac{n}{(1+b(n-1))} \qquad (5)$$

2.4 Prefetch Cache

Among various prefetch techniques, this paper assumed that the instruction fetcher has prefetch buffers to be able to predict the next starting address with one branch instruction in prefetch buffer. Fig. 4. is fetching example of prefetch cache.

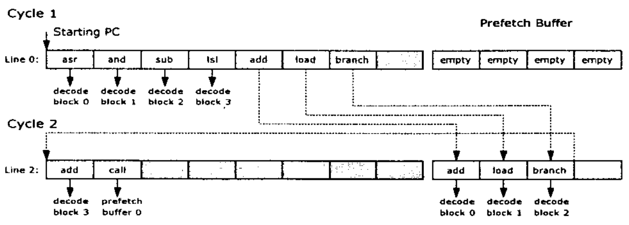

Fig. 4. Prefetch Fetching Example.

The expected fetch rate is as follows.

$$F_{prefetch}(n, b, m) = \frac{m-n}{m}L(n, b) + \frac{n}{m}L(n, b)$$
$$= L(n, b) = \frac{1-(1-b)^n}{b} \qquad (6)$$

2.5 Interleaved Cache

Fig. 5. Interleaved Fetching Example.

Conte introduced the concept of memory banks into the cache structure[2]. Instruction fetcher can predict next two target addresses from current PC using multiple branch predictor. With these two predicted PCs, individual two simple caches are referenced simultaneously.

$$F_{interleaved}(n,\ b) = 2F_{simple}(n,\ b) \quad (\le n) \qquad (7)$$

3. Analytical Models Considering Cache Misses and Branch Prediction Misses

3.1 Simple Cache and Extended Cache

If an instruction cache miss occurs or a branch prediction is wrong, instructions can not be fetched any more during the miss penalty. We will consider additionally wasted cycles due to cache misses and branch prediction misses. Let B be the frequency of branch instructions in program, R_C be cache miss rate, R_B be branch prediction miss rate, P_C be cache miss penalty, P_B be branch prediction miss penalty, and C_i

be additionally wasted cycles in case *i*.

Table 1. Fetch Processing Considering Various Factors.

Sequential Access		Non Sequential Access					
CH	CM	CH			CM		
		BPH	BPM		BPH	BPM	
			CH	CM		CH	CM
-	Consider P_C	-	Consider P_B	Consider P_C, P_B	Consider P_C	Consider P_C, P_B	Consider P_C, P_B
case(1)	case(2)	case(3)	case(4)	case(5)	case(6)	case(7)	case(8)

CH: Cache Hit, CM: Cache Miss,
BPH: Branch Prediction Hit, BPM: Branch Predictin Miss.

Table 1 shows the instruction fetch processing according to cache access patterns, cache miss occurrences, and branch prediction miss occurrences to compute additionally wasted cycles in each case *i*. For example, In case 2, instruction fetcher sequentially access the next cache line but the requested instruction cannot be founded(*cache miss*). Until the instruction is replaced from sub-memory, cache cannot serve fetch processing any more. Therefore, we must consider wasted cycles due to cache miss penalties corresponding to instructions in case 2.

$$C_2 = (1-B)\cdot R_C \cdot P_C \qquad (8)$$

Additionally cycles can be calculated each case.

$$C_4 = B\cdot(1-R_C)^2\cdot R_B\cdot P_B$$
$$C_5 = B\cdot(1-R_C)\cdot R_B\cdot R_C\cdot(P_C+P_B)$$
$$\cdots$$
$$C_{total} = \sum_{i=1}^{8} C_i = C_2 + C_4 + C_5 + C_6 + C_7 + C_8 \qquad (9)$$

C_{total} means additionally wasted total cycles for executing whole instructions of program. Thus, equations (4) and (5) can be modified as follows.

$$F_{modified} = \frac{F_{original}}{1+C_{total}} \qquad (10)$$

3.2 Prefetch Cache

In case of prefetch cache, different fetch rates must be applied according to each case. The fetch rate of prefetch cache, equation (6), just can be applied to instructions corresponding to case(1) and case(3).

$$\begin{aligned} F_{case(1)+case(3)} &= [(1-B)(1-R_C) \\ &\quad + B(1-R_C)(1-R_B)]\cdot F_{prefetch} \\ &= W\cdot F_{prefetch} \end{aligned} \qquad (11)$$

However, if an instruction cache miss occurs or a branch prediction is wrong, the fetch rate of extended cache must be used because only instructions in prefetch buffer can be fetched.

$$F_{case(2)+\sum_{i=4}^{8} case(i)} = (1-W)\cdot\frac{F_{extended}}{1+C_{total}} \qquad (12)$$

The modified fetch rate of prefetch cache is as follows.

$$F_{modified} = W\cdot F_{prefetch} + (1-W)\cdot\frac{F_{extended}}{1+C_{total}} \qquad (13)$$

3.3 Interleaved Cache

Table 2 and table 3 show the instruction fetch processing in interleaved cache like table 1 and mean sequential cache access and non sequential cache access respectively.

Table. 2 Fetch Processing Considering Various Factors of sequential cache access in interleaved cache.

Bank 0,1 CH		Bank 1 CM	Bank 0 CH	
Bank 1 BPH	Bank 1 BPM		Bank 1 BPH	Bank 1 BPM
-	-	-	Consider P_C	Consider P_C
Case(1)	Case(2)	Case(3)	Case(4)	Case(5)

Table. 3 Fetch Processing Considering Various Factors of non-sequential cache access in interleaved cache.

Bank 0,1 CH			Bank 1 CM		Bank 0 CM		
Bank 0,1 BPH	Bank 1 BPM	Bank 0 BPM	Bank 0 BPH	Bank 0 BPM	Bank 0,1 BPH	Bank 1 BPM	Bank 0 BPM
-	-	Consider P_B	-	Consider P_B	Consider P_C	Consider P_C	Consider P_C, P_B
Case(6)	Case(7)	Case(8)	Case(9)	Case(10)	Case(11)	Case(12)	Case(13)

In case of interleaved cache, all instruction fetch patterns are classified into four groups. First group(W_{I}) is a set of cases which apply only simple fetch rate, second one(W_{II}) is a set of cases which apply simple fetch rate considering additionally wasted cycles, third one(W_{III}) is a set of cases which apply interleaved fetch rate, and the last one(W_{IV}) is a set of cases which apply interleaved fetch rate considering additionally wasted cycles.

$$\begin{aligned} W_{\mathrm{I}} &= W_{2,3,7,9} \\ &= (1-B)(1-R_C)^2R_B + (1-B)(1-R_C)R_C \\ &\quad + B(1-R_C)^2(1-R_B)R_B + B(1-R_C)R_C(1-R_B) \\ W_{\mathrm{II}} &= W_{5,12} = (1-B)\cdot R_C\cdot R_B + B\cdot R_C\cdot(1-R_B)\cdot R_B \\ W_{\mathrm{III}} &= W_{1,6} \\ &= (1-B)(1-R_C)^2(1-R_B) + B(1-R_C)^2(1-R_B)^2 \\ W_{\mathrm{IV}} &= W_{4,8,10,11,13} = 1 - \sum_{i=1}^{3} W_i \end{aligned} \qquad (14)$$

The subscripts of second term mean corresponding cases in table 2, 3. If we consider all conditions, the fetch rate of interleaved cache can be modified as follows.

$$\begin{aligned} F_{modified} &= W_{\mathrm{I}}\cdot F_{simple} + W_{\mathrm{II}}\cdot\frac{F_{simple}}{1+C_5+C_{12}} + W_{\mathrm{III}}\cdot F_{interleaved} \\ &\quad + W_{\mathrm{IV}}\cdot\frac{F_{interleaved}}{1+C_4+C_8+C_{10}+C_{11}+C_{13}} \end{aligned} \qquad (15)$$

4. Simulations and Analyses

4.1 Simulator

In our simulations, we employ the basic fetching model as shown in Fig. 1. For precise modeling of fetch processing, the simulator is coded with C++. Benchmark programs are compiled using GNU C/C++ compiler with standard option in the SPARC workstation. To get instruction execution traces, we used Spy program which is included Spa package [3].

In the simulation model, if cache line include a branch instruction, all instructions after it are invalidated and if cache miss or branch prediction miss occurs, the cache cannot serve fetch processing any more until the miss is resolved. Parameters such as the frequencies of branch instructions *B*, cache miss rate R_C, and branch prediction miss rate R_B are acquired from simulations. Table 4 shows four benchmark programs used to generate instruction traces and values of the frequencies of branch instructions.

Table 4. Benchmark Programs used to generate Instruction Traces.

Program	GCC	Compress	Li	Tomcatv
The Frequency of Branch Instruction (%)	13.333	10.070	14.493	0.0455

4.2 Analytical Models and Simulations

The used cache models has 256 entries and 8 cache line size and uses direct-mapped placement method. we assume that R_C is 10%, R_B is 5%, P_C is 5 cycles, and P_B is 2 cycles. Fig. 6 is result of Li benchmark program and Fig. 7. is that of Tomcatv benchmark program. Simulation results showed that the proposed model can estimate the expected instruction fetch rate accurately within 10% differences in most cases.

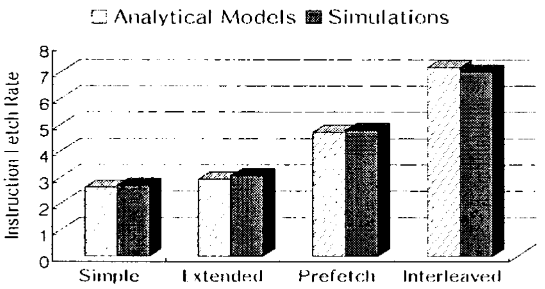

Fig. 6. The Results of Li Benchmark Program.

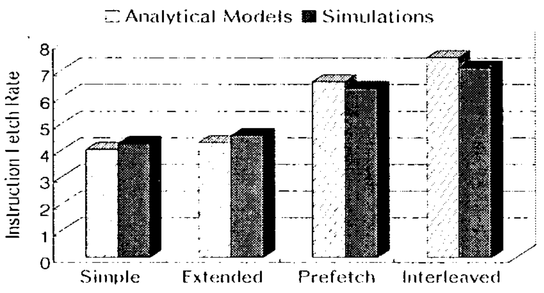

Fig. 7. The Results of Tomcatv Benchmark Program.

Fig. 8 shows the effects of the frequency of branch instructions on Li benchmark program. Fig. 9 shows the effects of cache miss rate and branch miss prediction rate and Fig. 10 shows that of cache miss penalty and branch prediction miss penalty respectively on GCC benchmark program. Simulations reveal that the decrease of instruction fetch rate is more affected by increase of cache miss rate rather than that of branch prediction miss rate. Similarly, the effects of cache miss penalties are bigger than the effects of branch prediction miss penalties. Thus we can know that cache misses must be minimized in designing new superscalar system.

5. Conclusion

To predict the performance of cache memories, most previous researches have concentrated on simulation based approach. But theses methods cannot explain precisely the causes of performance losses. This paper presents an analytical model to estimate the expected instruction fetch rate. The proposed model takes into account various kind of architectural parameters such as branch probability, cache miss rate, branch prediction miss rate, and etc.. The proposed model can explain practical instruction fetch rates according to various cache structure types. To prove the correctness of the proposed model, we performed extensive simulations and compared the results with those of the analytical model. Simulation results showed that the proposed model can estimate the instruction fetch rate accurately within 10% error in most cases. From analytical models, we fount that the performance of a cache fetch processing is affected more severely by the cache misses rather than branch prediction misses.

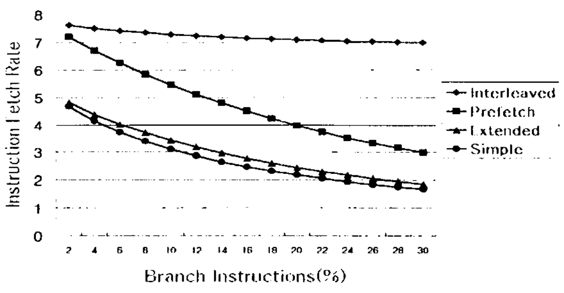

Fig. 8. The Effects of Branch Instructions.

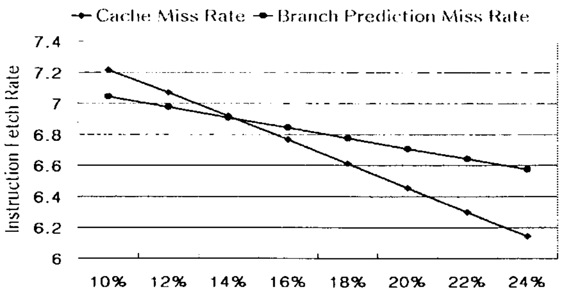

Fig. 9. The Effects of Cache Miss Rate and Branch Prediction Miss Rate.

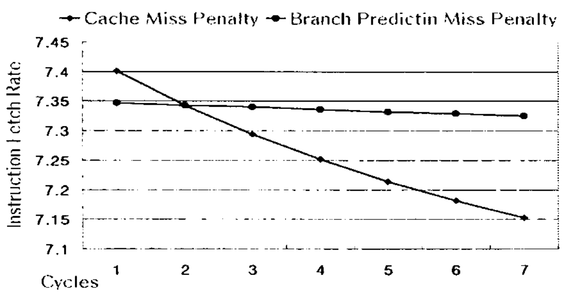

Fig. 10. The Effects of Cache Miss Penalty and Branch Prediction Miss Penalty.

References

[1] Steven Wallace and Nader Bagherzadeh, "Modeled and Measured Instruction Fetching Performance for Superscalar Microprocessors," *IEEE Trans. Parallel and Distributed Systems*, vol. 9, no. 6, pp. 570-578, Jun. 1998.

[2] Thomas M. Conte, Kishore N. Menezes, Patrick M. Mills, and Burzin A. Patel, "Optimization of Instruction Fetch Mechanisms for High Issue Rates," *Proc. 22nd Ann. Int'l Symp. Computer Architecture*, pp. 333-344, June 1995.

[3] Gordon Irlam, "Spa" Personal Communication 1995. http://www.base.com/gordoni/spa/cat1/spy.1