EISC processor의 속도 향상을 위한 pipelineing 최적화

손 무창*, 김 인수*, 민 형복*, 이 영걸**
성균관대학교 정보통신공학부*, 대림대학 컴퓨터정보계열**

EISC pipelineing optimizations for processor speed improvements

Mu Chang Son*, Insoo Kim*, Hyoung Bok Min*, Young-Geol Lee**
School of Information&Communication Engineering, Sungkyunkwan University*
Division of Computer Science and Information, Daelim College**

Abstract - Currently the quarter prediction giga it is used SE3208 from EISC ISA [1]] where it does in base. But the prediction which is perfect is difficult improved Pipeline structures and PC the structure which is not Delay to add it decided. Even PC and IF/ID blocks, the area and expense were added, but Bubble without it will be able to control Conditional Branch doors and the possibility of decreasing a help in processor performance improvements.

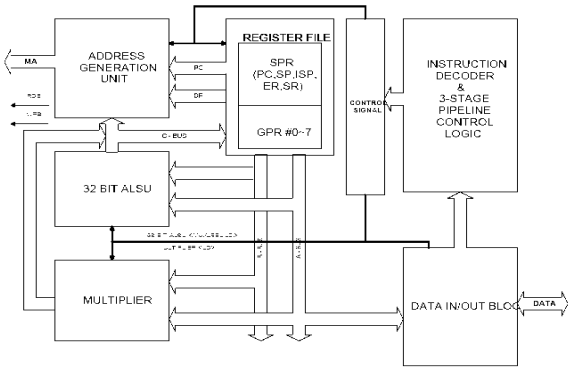
1. 서 론

현재 EISC 아키텍쳐를 기반으로 하는 SE3208에서는 Conditional Branch 문에서 분기예측기[2]가 사용되고 있다. 그러나 분기예측기에서는 완벽한 예측은 어렵다. 본 논문에서는 기존의 1bit Last_Time Branch Predictor와 분기예측테이블(Branch Prediction Table) 방식[3]을 벗어나 Conditional Branch 문에서 Bubble이 완벽히 발생하지 않는 구조를 제안한다. 제안된 ID/IF Block을 두개를 사용하고 Program Counter를 두 개를 사용해 Delay를 없앨 수 있다.

2. 본 론

2.1 SE3208

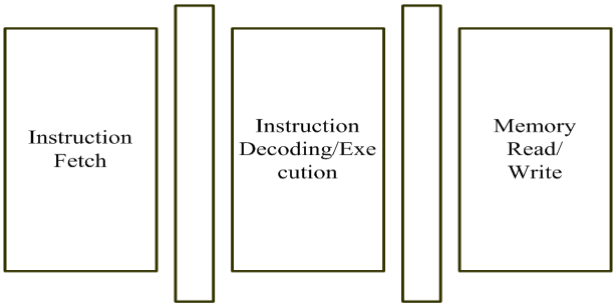
SE3208은 32 bit simple EISC microprocessor로 30-50 MIPS 이하의 performance를 필요로 하는 대부분의 embedded market에 적합하도록 개발된 제품이다. hardware가 간단하고, code density가 높고, performance가 높은 장점을 가지고 있다.[4]
그림 1에서 알 수 있듯이 32 Bit ALSU, General Purpose Registers, Special Purpose Register, instruction set architecture, Extendable instruction set 등을 갖추고 있다.



<그림 1> SE3208의 Block Diagram[5]

2.2 SE3208에서의 3 Stage pipeline 구조

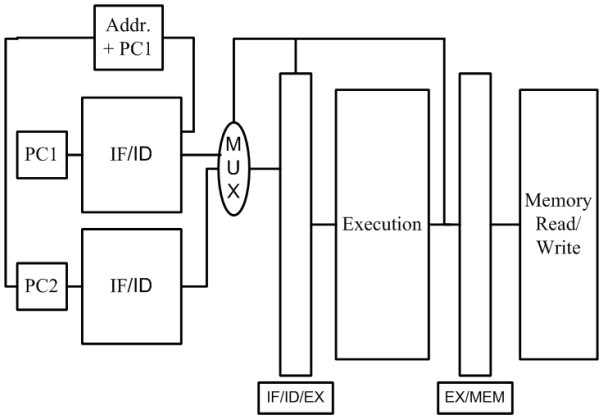
3 stage pipeline의 구조를 가지고 있으며, Instruction Fetch Cycle, Instruction Decoding/Execution, Memory Read/Write Cycle 으로 구성된다.



<그림 2> 기존 SE3208의 Pipeline 구조

2.3 제안하는 방법

앞서 본 SE3208의 pipeline은 Instruction Fetch Cycle, Instruction Decoding/Execution, Memory Read/Write Cycle의 3단계로 구성되는데, Conditional Branch 문에서 Bubble이 완벽히 발생하지 않는 방법을 위해 Instruction Fetch / Decoding Cycle, Execution / ALU, Memory Read / Write Cycle의 3단계로 바꾸면서, Program Counter를 하나 더 추가 하면 Bubble을 완벽히 없앨 수 있다.



<그림 3> 제안하는 Pipeline 구조

그림 3은 PC1에서 프로그램을 수행하다가 Conditional Branch 문을 만나면 ID단계에서 Address 값과 PC1값을 더한 값을 PC2, PC1+2의 값을 PC1로 만들어 동시에 IF로 들어간다. 그 시간 Execution에서는 Conditional Branch문을 수행하게 된다. Execution에서 Conditional Branch 결과가 나왔을 때, 각각의 Instruction Decoding 결과가 MUX로 들어간다. Execution에서 나온 결과와 Conditional Branch문을 인식하게 하는 signal을 생성해서 나온 값을 Control값으로 MUX에서 지나갈 signal을 구별해낸다. MUX에서 선택된 PC값을 다시 PC1으로 보내줘 이어지는 프로그램을 수행한다.
다시 말하면, Excution단계에서 Branch 문을 수행하고 있을 때, targeting 된 Address와 PC + 2 인 Address를 동시에 IF/ID stage로 들어가 Branch의 결과를 기다려 결과를 Control 해서 MUX를 통해 선택한다.

<표 1> MUX에서의 Control Signal 및 사용되는 PC

		Control signal	사용되는 PC
Conditional Branch 문	taken	11	PC2
	no taken	10	PC1
일반 문		0X	PC1

표 1에서는 code와 Conditional Branch문 수행결과에 따른 Control signal 과 사용되는 Program Counter를 나타낸다.

3. 분 석

제안된 구조를 통해 Conditional Branch문이 있더라도 조건계산이 끝날 때까지 기다릴 필요 없이 연속적으로 프로그램이 동작이 가능해진다. Bubble이 발생하지 않아 performance를 개선할 수 있다.

또한 Instruction Fetch와 Decoding 단계를 한 cycle에 묶어버림으로써 시간이 오래 걸리는 Execution중에서 ALU의 부담을 덜어 줄 수 있다.

하지만 ID/IF의 superscalar로 인한 chip size 증가는 단점으로 나타난다.

4. 결 론

본 논문에서는 제안하는 구조의 기존의 분기예측기에서 처럼 Condition을 예측하는 것이 아니라 ID/IF의 superscalar와 두 개의 PC를 이용해서 Conditional Branch를 분석해 지연 없는 파이프라인 구조로 개선하였다. 분석 결과 제안된 구조에서 Delay는 발생하지 않고, Pipelining에서의 시간분배가 적절히 되어 processor의 performance가 개선된다.

[참 고 문 헌]

[1] H. Lee, P. Beckett, B. Appelbe, High-performance extendable instuction set computing, Proc. of ACSAC, pp. 89-94, Jan. 2001.

[2] S. Lee, I. Kim, L. Choi, Branch Predictor Design and Performance Estimation for High Performance Embedded Microprocessor Oct. 2003.

[3] S. Shin, R. Choi, Implementation of a Branch Predictor and It's Cost Per Performance Analysis for a High Performance Embedded Microprocessor. 2003.

[4] Advanced Digital Chips Inc,
http://www.adc.co.kr/Product/product_eisc_07.asp

[5] ADchips, SE3208 Core Manual, Extendable Instruction Set Computer pp 11, Nob. 2003.