Mai Vu

Final Exam Report

Information Technology

Modern Computer Architecture

10th December 2019

1 Branch Prediction

∘ *Explain what is branch prediction.*

Let's start with a good old example. Imagine you are a full-time employee for a train company in the 1700-1800s before the invention of radio communication in the last decades of the 1800s. Your job is to switch a junction for trains to go the right path. As you cannot talk to the train drivers before the train approaches the junction, you have to stop every train to ask for the direction in order to correctly shift the junction. You come up with an idea: instead of doing as the manual instructions said, you guess the direction the train would go, wave to the train driver saying this is the right path and let them go without stopping. Of course, if that is a wrong guess, the train has to make a U-turn which takes more time than normal. Although just imagine if you can choose the right path with a significantly high probability, the trains will go on and on, no time to be wasted. That comes the idea of branch prediction.

Stage/Cycle	1	2	3	4
Fetch	C	I1	I2	I3
Read		C	I1	I2
ALU			C	I1
Memory				C
Write				

Figure 1. A 5-stage pipeline example.

Let's apply this idea into a classical 5-stage pipeline and for example, a conditional instruction is about to be executed as the above figure. The condition (marked as C) is fetched in the first cycle. In the second cycle, it moves to the read stage. Remember the result of the condition is unknown until the end of the ALU stage. The processor might wait until that stage to fetch further instructions based on the result; however, it is time-wasting, similar to our train employee example, stopping trains. Instead, the processor fetches instructions which it guesses to be the right one, letting no parts being idle. This is a simple example of branch prediction. After the third stage and the result is calculated,

if the following instructions are truly right, the following instructions are immediately executed without delays. If not, the wrong one will be discarded (pipeline flush) and the right one will be fetched; the wasted time is 2 cycles. Compared with no branch prediction, the processor still has to wait for 2 cycles. Therefore, it chooses the better solution: be able to save time in some cases. With nowadays technology, branch prediction can have accuracy up to 90%, meaning it actually saves a lot of time.

◦ *What different strategies can be used in prediction?*

Static and dynamic are 2 types of branch prediction. The difference between those is static branch prediction choose 1 state every time, meaning it does not need to store data of previous executions to change prediction like the dynamic one. There are many different approaches based on the type such as always not-taken, global branch prediction and local branch prediction, etc. I name 2 simplest examples for each type and simply describe their operation through the below table. Overall, the static branch prediction results in more wasted time compared with dynamic prediction but it does improve the total execution time in general.

Name	Type of branch prediction	Operation
Always-taken	Static	As the name described, the predict taken is always chosen.
Backward taken forward not taken	Static	The prediction goes back to the backward code instead of the upcoming code.
1-bit	Dynamic	The prediction is depended on the previous behavior or the latest related branch. There is 1 bit reserved for taken (1) or not-taken (0) decision.
2-bit	Dynamic	Similar to 1-bit branch prediction, there are 2 bits intended for predictions: strongly not-taken (00), weakly not-taken (01), weakly taken (10) and strongly taken (11).

Table 1. Table of 4 branch prediction examples.

◦ *Explain why branch prediction is very important in a modern processor.*

Firstly, conditional branches account for about one-fifth of all instructions. Besides, modern processors can have up to 20 pipeline-stage, meaning more cycles penalty and superscalars are used, meaning more instructions are executed in a cycle. Those would result in greater wasted time if the branch prediction is not performed and if the branch accuracy is low. In conclusion, branch prediction is very important in modern processors. More algorithms such as perceptron (single neural layer) are applied in order to achieve higher accuracy.

2 Direct and Set Associative Mapping

The direct mapping and set associative mapping are 2 of 3 types of cache placement policies. The other type is fully associative mapping. In direct mapping, a block in the memory can be mapped into the cache line following 1 rule: the memory index's least significant bits equal to the cache line index, meaning blocks have a fix position in the cache. While the fully associate mapping allows more freedom in mapping, meaning any block can be placed into any free line in the cache. The set associative mapping is a combination of the 2 above types. In detail, its cache is divided into sets in which a set has a number of lines. Memory blocks are placed into a set using direct mapping and into a line in a set using fully associative mapping.

Therefore, direct mapping and set associative mapping share the rule in common. In fact, a direct mapping is a 1-way associative mapping. There are differences as well. Direct mapping is easier to implement, while set associative mapping requires more complex cache architecture, for example, tag parallel checking. Direct mapping does not need cache replacement algorithms since if the conflict miss happens, the new block will be replaced at the same location every time. Because there are cache positions that a block can be replaced in set associative mapping, it needs cache replacement policies. Overall, direct mapping results in more conflict misses and lower hit rate compared to set associative one. Therefore, set associative mapping is a better technique. Considering this scenario, in the direct-mapped cache, when 2 blocks that have the same position mapping in the cache are repeatedly referenced, the same cache line is replaced continually for 2 blocks to swap places. This increases the cache misses and also cause thrashing. While in more or equal 2-way set associative, this would not happen. In real life, almost all modern processors are implemented with set associate mapping instead of the direct one.

3 Terms Explanation

3.1 Single Instruction, Multiple Data

As described in the name, single instruction multiple data or SIMD describes the processing performing only 1 instruction for multiple sets of data at the same time. A simple example is adding 2 input arrays; there is an addition operation for pairs of input values and the process is adding pairs in parallel. SIMD operations include arithmetic and some other operations. SIMD instructions are implemented in most of nowadays processors in order to enhance multimedia and scientific performance.

3.2 Control Hazard

Control hazard happens when there is an instruction in the pipeline that disrupt the control flow or the order of instructions to be executed. That instruction could be an unconditional branch, conditional branch or function call, etc. For example, the pipeline does not know what is the next instruction of a jump function (unconditional instruction) or an if-else statement (conditional instruction) until those instructions get to later stages for results to be calculated. One of the solutions is the branch (target) prediction. The pipeline will continue to fetch predicted instructions. Later on, if they are wrong, they have to be discarded. Otherwise, the pipeline continues the flow and no time is wasted.

3.3 Cache Coherency

Nowadays computer architecture has multiple central processing units (CPUs) to boost performances. Each CPU has its separate cache to ensure the speed of processing data. It comes with a problem; since those processors run in parallel, independent caches might hold different values due to no synchronization is done. Therefore resulting in inconsistent values. Cache coherence indicates the connection for data to be updated through changes on multiple caches. Directory and snoopy protocols are methods to maintain cache coherence and keep data consistent.

3.4 Direct Memory Access

Direct memory access (or DMA) is a method that provides an input/output device the right to independently access main memory without the central processing unit involvement. DMA controller is in charge of this process. The reason for this feature is that some data does not need to be processed or the process can be done by other hardware such as graphic cards, sound cards. DMA allows the job to be divided, reduces the workload for the CPU and therefore, significantly improves the overall performance.

4 Processors Testing

The below formula is used to calculate the execution time (also called CPU time). Results for tests of processors are displayed in table 2. (Note: penalty = pipeline stage – 1 and mispredict frequency = 1)

$$CPU\ time = Instruction\ count \times Cycles\ per\ instruction \times Clock\ cycle\ time$$

Ex	Test 1	Test 2	Test 3
A	≈ 0.020s	≈ 0.0184s	≈ 0.016s
B	≈ 0.014s	≈ 0.0125s	≈ 0.010s
C	≈ 0.015s	≈ 0.0128s	≈ 0.009s

Table 2. Execution time calculation for 3 tests for 3 processors.

SUF is calculated using the below formula. Table 3 shows the SUF of 3 processors' tests.

$$SUF = \frac{1}{1 + Branch\ frequency \times Mispredict\ frequency \times Penalty}$$

SUF	Test 1	Test 2	Test 3
A	≈ 0.769	≈ 0.833	≈ 0.970
B	≈ 0.690	≈ 0.770	≈ 0.957
C	≈ 0.571	≈ 0.667	≈ 0.930

Table 3. SUF calculation for 3 tests for 3 processors.

5 Security Measures

◦ *Which of the security measures in the article are implemented in the hardware and what parts of the CPU are in charge of enforcing the security measures?*

PS4 uses a technique called executable space protection as security measures. In detail, this method marks the memory sections into executable and non-executable zones. There is a Memory Management Unit or MMU inside the custom AMD x86-64 CPU providing no-execute bit (NX bit) for marking.

Additionally, as mentioned in part 1 and 2 of the article, cryptography feature is also implemented in the hardware. Encryption is controlled by a part of PS4's CPU called Secure Asset Management Unit (SAMU). In details, SAMU is a hardware security module or HSM which has secure cryptoprocessor chips. SAMU was nicely developed by Semiconductor Company that it is extremely difficult to hack.

◦ *How malicious code can be executed with ROP even when the security measures prevent injecting executable code in to the system?*

ROP or Return-Oriented Programming is a technique that exploits the vulnerable system. Using this method, despite the security defenses, attackers can execute injected code.

ROP is a high-level update of the stack smashing attack. In a normal stack smashing attack, the call stack is manipulated after exploiting the buffer overflow. The process is to simply input more data onto the stack in order to overwrite the return address which is used to return to the caller. As a result, the control flow is now redirected to the wrong address or the malicious instructions.

Later on, the hardware is updated to prevent this threat; a technique called executable space protection. This allows only instructions that are not in the user-writeable section or the non-executable in the memory to be executed. However, ROP overcomes this defense. It takes advantage of some instructions in the executable area of the memory or gadgets. So at runtime, the compiler is manipulated and malicious written code which has been marked as executable is executed. In conclusion, using ROP, malicious instructions still can be executed despite security measures.