Computer Architecture

~ Homework #4 ~

2013210111 남세현

4.14 This exercise is intended to help you understand the relationship between delay slots, control hazards, and branch execution in a pipelined processor. In this exercise, we assume that the following MIPS code is executed on a pipelined processor with a 5-stage pipeline, full forwarding, and a predict-taken branch predictor:

```
        Lw      r2, 0(r1)
label1: beq     r2, r0, label2 # not taken once, then taken
        lw      r3, 0(r2)
        beq     r3, r0, label1 # taken
        add     r1, r3, r1
label2: sw      r1, 0(r2)
```

4.14.1 [10] <§4.8> Draw the pipeline execution diagram for this code, assuming there are no delay slots and that branches execute in the EX stage.

1 LW	IF	ID	EX	MEM	WB										
2 BEQ		IF	ID	STALL	EX	MEM	WB								
3 LW			STALL	STALL	STALL	IF	ID	EX	MEM	WB					
4 BEQ							IF	ID	STALL	EX	MEM	WB			
Stall								stall	stall	Stall					
2 BEQ											IF	ID	EX	MEM	WB
Stall												stall	stall		
6 SW														IF~	

4.14.2 [10] <§4.8> Repeat 4.14.1, but assume that delay slots are used. In the given code, the instruction that follows the branch is now the delay slot instruction for that branch.

1 LW	IF	ID	EX	MEM	WB										
2 BEQ		IF	ID	STALL	EX	MEM	WB								
3 LW			IF	STALL	ID	EX	MEM	WB							
4 BEQ					IF	ID	STALL	EX	MEM	WB					
5 ADD						IF	STALL	ID	FLUSH						
6 SW								IF	FLUSH						
2 BEQ									IF	ID	EX	MEM	WB		
3 LW										IF	ID	FLUSH			
4 BEQ											IF	FLUSH			
6 SW												IF	ID	EX	MEM/WB

4.14.3 [20] <§4.8> One way to move the branch resolution one stage earlier is to not need an ALU operation in conditional branches. The branch instructions would be “bez rd,label” and “bnez rd,label”, and it would branch if the register has and does not have a zero value, respectively. Change this code to use these branch instructions instead of beq. You can assume that register R8 is available for you to use as a temporary register, and that an seq (set if equal) R-type instruction can be used.

```
        Lw      r2, 0(r1)
        seq     r8, r2, r0
label1: bez     r8, label2
        lw      r3, 0(r2)
        seq     r8, r3, r0
        bez     r8, label1
        add     r1, r3, r1
label2: sw      r1, 0(r2)
```

5.2 Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 32-bit memory address references, given as word addresses.

3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253

5.2.1 [10] <§5.3> For each of these references, identify the binary address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

Block size is 1 word.

Block count is 16

So, 4 index bit need(2^4 = 16)

28 bit for tag.

Binary Address와 Tag 앞의 0은 생략.

Decimal addr	Binary address	Tag	index	h/m
3	0000 0011	0000	0011	M
180	1011 0100	1011	0100	M
43	0010 1011	0010	1011	M
2	0000 0010	0000	0010	M
191	1011 1111	1011	1111	M
88	0101 1000	0101	1000	M
190	1011 1110	1011	1110	M
14	0000 1110	0000	1110	M
181	1011 0101	1011	0101	M
44	0010 1100	0010	1100	M
186	1011 1010	1011	1010	M
253	1111 1101	1111	1101	M

5.2.2 [10] <§5.3> For each of these references, identify the binary address, the tag, and the index given a direct-mapped cache with two-word blocks and a total size of 8 blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

8 block size = 2^3, so index = 3 bit,

Decimal addr	Binary address	Tag	index	h/m
3	0000 0011	0000	001	M
180	1011 0100	1011	010	M
43	0010 1011	0010	101	M
2	0000 0010	0000	001	H
191	1011 1111	1011	111	M
88	0101 1000	0101	100	M
190	1011 1110	1011	111	H
14	0000 1110	0000	111	M
181	1011 0101	1011	010	H
44	0010 1100	0010	110	M
186	1011 1010	1011	101	M
253	1111 1101	1111	110	M

5.2.3 [20] <§§5.3, 5.4> You are asked to optimize a cache design for the given references. There are three direct-mapped cache designs possible, all with a total of 8 words of data: C1 has 1-word blocks, C2 has 2-word blocks, and C3 has 4-word blocks. In terms of miss rate, which cache design is the best? If the miss stall time is 25 cycles, and C1 has an access time of 2 cycles, C2 takes 3 cycles, and C3 takes 5 cycles, which is the best cache design?

n-word blocks에서 n이 높을수록 miss rate가 낮다. 2^n배 더 좋다고 할 수 있다.

C3을 기준으로 A% 미스라고 하면 C2는 2A, C1은 4A의 미스 확률을 가진다.

C3 : A * 25 + 5 * (1 – A) = 5 + 20A

C2 : 2A * 25 + 3 * (1 – A) = 3 + 47A

C1 : 4A * 25 + 1 * (1 – A) = 1 + 99A

4-wrod blocks 캐쉬에서 8 word의 데이터가 미스될 확률은 1/2이다. 그러므로

C3 = 15, C2 = 14.75, C1 = 13.375 으로 C1이 가장 좋은 성능을 가진다.

There are many different design parameters that are important to a cache's overall performance. Below are listed parameters for different direct-mapped cache designs.

Cache Data Size: 32 KiB

Cache Block Size: 2 words

Cache Access Time: 1 cycle

5.2.4 [15] <§5.3> Calculate the total number of bits required for the cache listed above, assuming a 32-bit address. Given that total size, find the total size of the closest direct-mapped cache with 16-word blocks of equal size or greater. Explain why the second cache, despite its larger data size, might provide slower performance than the first cache.

32KiB = 2 ^ 15 byte = 2 ^ 15 / 4 = 2^13 words. Index bits = 13 – 1 = 12

Tag bit = 32 – 12 = 18

```
(2^(index bits)) * ((tag bits)+(valid bits)+(data size))
```

2 ^ 12 * (18 + 1 + 64) = 339968 bits is total number of bits.