COMPSCI 2GA3 Fall, 2021

Assignment/Homework 4, Nov. 21st 2021
Assignment due date: Dec. 8th, 23:59:59.
Note: Please work on this assignment individually. Students copying each other's answer will get a zero and will perform poor on midterm and final.

Written Exercises

Complete the following questions from *Computer Organization and Design: The Hardware Software Interface: Computer Organization and Design The Hardware/Software Interface: RISC-V Edition*

1. Exercise 4.27.1-3 (5 Marks) Problems in this exercise refer to the following sequence of instructions, and assume that it is executed on a five-stage pipelined datapath:

```
add x15, x12, x11
ld  x13, 4(x15)
ld  x12, 0(x2)
or  x13, x15, x13
sd  x13, 0(x15)
```

   4.27.1 If there is no forwarding or hazard detection, insert `NOP`s to ensure correct execution.
   4.27.2 Now, if possible, change and/or rearrange the code to minimize the number of `NOP`s needed. You can assume register `x17` can be used to hold temporary values in your modified code.
   4.27.3 If the processor has forwarding, but we forgot to implement the hazard detection unit, what happens when the original code executes?

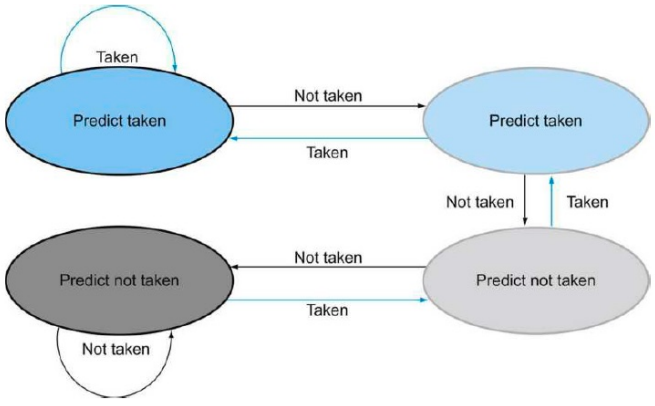
   Solution:

4.27.1
```
add   x15, x12, x11
nop
nop
ld    x13, 4(x15)
ld    x12, 0(x2)
nop
or    x13, x15, x13
nop
nop
sd    x13, 0(x15)
```


4.27.2 It is not possible to reduce the number of NOPs.

4.27.3 The code executes correctly. We need hazard detection only to insert a stall when the instruction following a load uses the result of the load. That does not happen in this case.

2. Exercise 4.29.1-3 (5 Marks) This exercise examines the accuracy of various branch predictors for the following repeating pattern (e.g., in a loop) of branch outcomes: T, NT, T, T, NT.
   4.29.1 What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?
   4.29.2 What is the accuracy of the 2-bit predictor for the first four branches in this pattern, assuming that the predictor starts off in the bottom left state from figure below (Figure 4.61) (predict not taken)?
   4.29.3 What is the accuracy of the 2-bit predictor if this pattern is repeated forever?



Solution:

4.29.1

Always Taken	Always not-taken
3/5 = 60%	2/5 = 40%

4.29.2

Outcomes	Predictor value at time of prediction	Correct of Incorrect	Accuracy
T, NT, T, T	0,1,0,1	I,C,I,I	25%

4.29.3 The first few recurrences of this pattern do not have the same accuracy as the later ones because the predictor is still warming up. To determine the accuracy in the "steady state", we must work through the branch predictions until the predictor values start repeating (i.e. until the predictor has the same value at the start of the current and the next recurrence of the pattern).

Outcomes	Predictor value at time of prediction	Correct of Incorrect (in steady state)	Accuracy
T, NT, T, T, NT	1st occurrence: 0,1,0,1,2 2nd occurrence: 1,2,1,2,3 3rd occurrence: 2,3,2,3,3 4th occurrence: 2,3,2,3,3	C,I,C,C,I	60%

3. Exercise 5.11.1-2 (5 Marks) This exercise examines the effect of different cache designs, specifically comparing associative caches to the direct-mapped caches from Section 5.4. For these exercises, refer to the sequence of word address shown below.

   `0x03, 0xb4, 0x2b, 0x02, 0xbe, 0x58, 0xbf,`
   `0x0e, 0x1f, 0xb5, 0xbf, 0xba, 0x2e, 0xce`

   5.11.1 Sketch the organization of a three-way set associative cache with two-word blocks and a total size of 48 words. Your sketch should have a style similar to Figure 5.18, but clearly show the width of the tag and data fields.

   5.11.2 Trace the behavior of the cache from Exercise 5.11.1. Assume a true LRU replacement policy. For each reference, identify

   - the binary word address
   - the tag
   - the index

- the offset
- whether the reference is a hit or a miss, and
- which tags are in each way of the cache after the reference has been handled

Solution:

5.11.1 Each line in the cache will have a total of six blocks (two in each of three ways). Th ere will be a total of $48/6 = 8$ lines.

5.11.2 T(x) is the tag at index x.

Word Address	Binary Address	Tag	Index	Offset	Hit/Miss	Way 0	Way 1	Way 2
0x03	0000 0011	0x0	1	1	M	T(1)=0		
0xb4	1011 0100	0xb	2	0	M	T(1)=0 T(2)=b		
0x2b	0010 1011	0x2	5	1	M	T(1)=0 T(2)=b T(5)=2		
0x02	0000 0010	0x0	1	0	H	T(1)=0 T(2)=b T(5)=2		
0xbe	1011 1110	0xb	7	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b		
0x58	0101 1000	0x5	4	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5		
0xbf	1011 1111	0xb	7	1	H	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5		
0x0e	0000 1110	0x0	7	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	
0x1f	0001 1111	0x1	7	1	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	T(7)=1
0xb5	1011 0101	0xb	2	1	H	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	T(7)=1
0xbf	1011 1111	0xb	7	1	H	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	T(7)=1
0xba	1011 1010	0xb	5	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=2 T(5)=b	T(7)=1
0x2e	0010 1110	0x2	7	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=2 T(5)=b	T(7)=1
0xce	1100 1110	0xc	7	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=2 T(5)=b	T(7)=c

4. Exercise 5.17.1-2 (5 Marks) There are several parameters that affect the overall size of the page table. Listed below are key page table parameters.

Virtual Address Size	Page Size	Page Table Entry Size
32 bits	8 KiB	4 bytes

5.17.1 Given the parameters shown above, calculate the maximum possible page table size for a system running five processes.

5.17.2 Given the parameters shown above, calculate the total page table size for a system running five applications that each utilize half of the virtual memory available, given a two-level page table approach with up to 256 entries at the 1st level. Assume each entry of the main page table is 6 bytes. Calculate the minimum and maximum amount of memory required for this page table.

Solution:
5.17.1: The tag size is 32-$\log_2(8192)$ = 32 - 13 = 19 bits. All five page tables would require 5 $\times$ (2^{19} $\times$ 4) bytes = 10 MB.
5.17.2: In the two-level approach, the 2^{19} page table entries are divided into 256 segments that are allocated on demand. Each of the second-level tables contains 2^{19-8} = 2048 entries, requiring 2048 $\times$ 4 = 8 KB each and covering 2048 $\times$ 8 KB = 16 MB (2^{24}) of the virtual address space.

We need 256*6B for first-level table and 2048 x 4 = 8KB for each second-level table, and we have 256 of those tables. So the minimum amount of memory required is 256*6B + 256*8KB
Maximum memory amount depends on the number of overhead bits we use for validity, replacement policy, write policy, etc