The University of Alabama in Huntsville
ECE Department
Homework Assignments
CPE 431/531 01/91/92
Fall 2015
Homework #8 Solution

5.12.3(15), 5.12.6(5), 6.4.1(10), 6.4.2(10), 6.7.1(10), 6.9.3(10), 6.18.2(10)

5.12 In this exercise, we will examine space/time optimizations for page tables. The following list provides parameters of a virtual memory system.

Virtual Address (bits)	Physical DRAM installed	Page Size	PTE Size (byte)
43	16 GiB	4 KiB	4

5.12.3 An inverted page table can be used to further optimize space and time. How many PTEs are needed to store the page table? Assuming a hash table implementation, what are the common case and worst case numbers of memory references needed for servicing a TLB miss?

The inverted page table will have as many entries as there are physical pages. The number of physical pages is 16 GiB/4 KiB = 4 MiB. The common case number of memory references is dependent on the hash function chosen. The worst case is 2^{22}.

The following shows the contents of a 4-entry TLB.

Entry-ID	Valid	VA Page	Modified	Protection	PA Page
1	1	140	1	RW	30
2	0	40	0	RX	34
3	1	200	1	RO	32
4	1	280	0	RW	31

5.12.6 What happens when an instruction writes to VA page 200?

Because VA page 200 is read only, an exception will be generated.

6.4 Consider the following piece of C code:

```
for (j = 2; j < 1000; j++)
  D[j] = D[j-1] + D[j-2];
```

The MIPS code corresponding to the above fragment is:

```
        addiu   $s2, $zero, 7992
        addiu   $s1, $zero, 16
Loop:   l.d     $f0, -16($s1)
        l.d     $f2, -8($s1)
        add.d   $f4, $f0, $f2
        s.d     $f4, 0($s1)
        addiu   $s1, $s1, 8
        bne     $s1, $s2, loop
```

Instructions have the following associated latencies (in cycles):

add.d	l.d	s.d	addiu
4	6	1	2

6.4.1 How many cycles does it take for all instructions in a single iteration of the above loop to execute?

2 x 6 + 1 x 4 + 1 x 1+ 1 x 2 + 1 x 1 = 20 cycles, assuming 1 cycle for bne, The two addiu take 4 cycles, but are not part of the loop iteration

6.4.2 When an instruction in a later iteration of a loop depends upon a data value produced in an earlier iteration of the same loop, we say that there is a loop carried dependence between iterations of the loop. Identify the loop-carried dependences in the above code. Identify the dependent program variable and assembly-level registers. You can ignore the loop induction variable `j`.

D[j] and D[j-1] are loop carried dependences, D[j] in the current iteration is the D[j-1] in the next iteration and D[j-1]- in the current iteration is the D[j-2] in the next iteration. These variables are stored in registers $f2 and $f4.

6.7 Consider the following portions of two different programs running at the same time on four processors in a symmetric multi-core processor (SMP). Assume that before this code is run, both x and y are 0.

Core 1: `x = 2;`
Core 2: `y = 2;`
Core 3: `w = x + y+ 1;`
Core 4: `z = x + y;`

6.7.1 What are all the possible resulting values of w, x, y, and z? For each possible outcome, explain how we might arrive at these values. You will need to examine all possible interleavings of instructions.

	w	x	y	z
1, 2, 3, 4	5	2	2	4
1, 2, 4, 3	5	2	2	4
1, 3, 2, 4	3	2	2	4
1, 3, 4, 2	3	2	2	2
1, 4, 2, 3	5	2	2	2
1, 4, 3, 2	3	2	2	2
2, 1, 3, 4	5	2	2	4
2, 1, 4, 3	5	2	2	4
2, 3, 1, 4	3	2	2	4
2, 3, 4, 1	3	2	2	4
2, 4, 1, 3	5	2	2	2
2, 4, 3, 1	3	2	2	2
3, 1, 2, 4	1	2	2	4
3, 1, 4, 2	1	2	2	2
3, 2, 1, 4	1	2	2	4
3, 2, 4, 1	1	2	2	2
3, 4, 1, 2	1	2	2	0
3, 4, 2, 1	1	2	2	0
4, 1, 2, 3	5	2	2	0
4, 1, 3, 2	3	2	2	0
4, 2, 1, 3	5	2	2	0
4, 2, 3, 1	3	2	2	0
4, 3, 1, 2	1	2	2	0
4, 3, 2, 1	1	2	2	0

6.9 Consider the following three CPU organizations:

CPU SS: A 2-core superscalar microprocessor that provides out-of-order issue capabilities on 2 function units (FUs). Only a single thread can run on each core at a time.

CPU MT: A fine-grained mutlthreaded processor that allows instructions from 2 threads to be run concurrently (i.e., there are two functional units), though only instructions from a single thread can be issued on any cycle.

CPU SMT: An SMT processor that allows instructions from 2 threads to be run concurrently (i.e., there are two functional units), and instructions from either or both threads can be issued to run on any cycle.

Assume we have two threads X and Y to run on these CPUs that include the following operations:

Thread X	Thread Y
A1 – takes 3 cycles to execute	B1 – take 2 cycles to execute
A2 – no dependences	B2 – conflicts for a functional unit with B1
A3 – conflicts for a functional unit with A1	B3 – depends on the result of B2
A4 – depends on the result of A3	B4 – nodependences and takes 2 cycles to execute

Assume all instructions take a single cycle to execute unless noted otherwise or they encounter a hazard.

6.9.3 Assume that you have 1 MT CPU. How many cycles will it take to execute these two thread? How many issue slots are wasted due to hazards?

FN1	FN2
A1	A2
A1	
A1	
B1	B4
B1	B4
A3	
A4	
B2	
B3	

It will take 9 cycles and 6 issue slots are wasted.

6.18 When performing computations on sparse matrices, latency in the memory hierarchy becomes much more of a factor. Sparse matrices lack the spatial locality in the data stream typically found in matrix operations. As a result, new matrix representations have been proposed.

One of the earliest sparse matrix representations is the Yale Sparse matrix Format. It stores an initial sparxe m x n matrix, M in row form using three one-dimensional arrays. Let R be the

number of nonzero entries in M. We construct an array A of length R that contains all nonzero entries of M (in left-to-right top-to-bottom order). We also construct a second array IA of length m + 1 (i.e., one entry per row, plus one). IA(i) contains the index in A of the first nonzero of element of fow i. Row I of the original matrix extends from A(IA(i)) to A(IA(I+1)-1). The third array, JA, contains the column index of each element of A, so it also is of length R.

```
Row 1 [1, 2, 0, 0, 0, 0]
Row 2 [0, 0, 1, 1, 0 ,0]
Row 3 [0, 0, 0, 0, 9 ,0]
Row 4 [2, 0, 0, 0, 0 ,2]
Row 5 [0, 0, 3, 3, 0 ,7]
Row 6 [1, 3, 0, 0, 0 ,1]
```

6.18.2 In terms of storage space, assuming that each element in matrix X is single precision floating point, compute the amount of storage used to store the Matrix above in Yale Sparse Matrix Format.

A is of length R (13 in this case) and IA is of length m + 1 (7) and JA is of length R (13). The total is 33.