Name Derek Hernandez

netID dh11a

(email not long Axxxxx number)

Assignment 5 CS 3339 – Spring 2019

Due: Friday, 4/12/18 @ 11:55pm

40 points (late until noon 11/10 -10 pts)

All submissions must be written in very neat handwriting and scanned (or typed) and submitted in PDF format to TRACS with the filename of Ax_netID.pdf. You may submit as many times as you like prior to the deadline; only the most recent submittal will be graded. All assignments must be submitted individually and reflect your own work; however, you are encouraged to work in groups and discuss the problems with your classmates.

1) [6 points] Dissassemble `00100100 : 152affe6` (written as 32-bit MIPS addr:instr in hex)

0001 0101 0010 1010 1111 1111 1110 0110

bne $t1, $t2

bne $t1, $t2, 1001000 9C

2) [6 points] Unroll the following loop four times and optimize the resulting instruction sequence. Assume the loop count is a multiple of four. Further assume the 5-stage MIPS pipeline with ID-stage branch resolution, and make sure that no bubbles need to be generated by the CPU. [Hint: There is a source dependency on $t0; be careful about bubbles from this!]

```
loop: sw    $a1, 0($t0)
      addi  $t0, $t0, 4
      bne   $t0, $a0, loop
```

```
sw   $a1, 0($t0)
sw   $a1, 4($t0)
sw   $a1, 8($t0)
addi $t0, $t0, 4
lw   $a1, 12($t0)
bne  $t0, $a0, loop
```

3) [6 points] Pipeline implementations introduce three types of hazards. The MIPS architecture avoids structural hazards by including all necessary hardware. Complete the following statements with respect to the other two types of hazards.

Data hazards exist when a prior instructions results are not available for a current instruction. These hazards are corrected by adding Stall cycles. In order to gain back some performance forwarding are implemented.

Control hazards exist when the next instruction executed is not located at PC+4. These hazards are corrected using flush cycles. In order to gain back some performance Branch Prediction is implemented.

4) [4 points] Assume a processor with a 2.0 GHz clock. The cache has a base cache access time (including hit detection) of 1 clock cycle, and L1 miss penalty of 10 cycles, and an L2 miss penalty of 40 cycles. Assume that 8% of read accesses to the L1 data cache miss and that 20% of read accesses to the L2 miss Compute the average number of clock cycles required for a load instruction.

referred to in class assign

$1 + .08[10 + .20(40)] = 2.44$ cycles/second

Clock cycle Cache miss

What is the average memory access time per load instruction in nanoseconds?

(AMAT)

2.44 cycles/second = $2.44 * 10^{-9}$ cycles/ns.

5) [4 points] A 4-way set associative write-back cache is implemented on a system with 32-bit words and addresses. It is organized with 16 words per block and has a total of 64 blocks. Show how the address bits A[31] to A[0] would map to the tag, index, word offset, and byte offset fields.

4 way
64 Blocks
16 Sets → 4 bits

2 bytes/word
16 words/Block
32 bytes/block

2 byte offset
4 word offset
2^4

Tag	Index	word off	Byte off
A[31] to A[10]	A[9] to A[6]	A[5] to A[2]	A[1] to A[0]

6) [4 points] Describe a pattern of memory access that would exhibit the types of locality listed below.

Temporal Locality

Copy newly accessed data

Pattern: Referenced in cluster time

Ex: A[1], A[2], A[3]

Spatial Locality

Copy neighboring data

Referenced closely together

A[0][0], A[0][1], A[0][3]

7) [4 points] Even without prefetching it is possible for the first-ever access to a word to hit in the cache. What allows this to happen? memory to cache

Since there is no Prefetching the only way to make a hit Possible would be if its part of the Previously accessed Cache block.

8) [6 points] Given a direct mapped cache with one word per block and four blocks with the following accesses, circle the cycles/addresses that are cache hits.

loop

0x0, 0x18, 0xC, 0x20, 0x18, 0x38, 0xC, 0x18, 0x20, 0x38

✓ ✗ ✗ ✗ ✗ ✗ ✗ ✗ ✓ ✗

00 01 10 11 00 01 10 11 00 01

1 words per b

4 b = blocks

00 - 11

C = 4 words

What is the hit ratio?

~~5/10 = 50%~~

X x18, x38, x38 loop

2/10 = 20%