CSEE W3827

Fundamentals of Computer Systems

Homework Assignment 5

Solutions

Profs. Stephen A. Edwards & Martha Kim

Columbia University

Due December 10th, 2012 at 5:00 PM
Turn in at CSB 469.

Write your name **and UNI** on your solutions

Show your work for each problem; we are more interested in how you get the answer than whether you get the right answer.

1. (25 points) Pipelined software execution.

   (a) Assuming a fully bypassed (i.e., both W-E and M-E operand forwarding), 5-stage MIPS pipeline with early branch resolution, indicate all of the stalls and operand forwarding that is used to execute the following ten instructions:

```
bnez $a0, tree_sum_recurse
addi $sp, $sp, -12  stall for branch
sw $ra, 0($sp)  forward $sp M to E
sw $s0, 4($sp)  forward $sp W to E
sw $s1, 8($sp)
move $s0, $a0
lw $s1, 0($s0)  forward $s0 M to E
lw $a0, 4($s0)  forward $s0 W to E
```

   (b) How many cycles will it take for this code to execute *completely* on the pipelined processor?

   8 instructions + 1 stall + 4 cycles to drain pipe = 13 cycles

   (c) Will this code run faster on a 100MHz single cycle processor or a 400MHz pipelined processor?

   8 cycles * 10ns/cycle = 80ns v. 13 cycles * 2.5ns/cycle = 32.5ns; it will be faster on the pipeline design.

2. (25 points) Assuming a pipeline with no bypassing, reorder the instructions such that they compute the same function, but require *no* pipeline stalls due to data hazards. Branch stalls are fine. You should reorder instructions as effectively as you can before inserting as few nops as possible. Of the nops you inserted, indicate which could be avoided with bypassing. FYI: This function takes a pointer to a string and a character, and counts how many times the character appears in the string.

```
count:
  addi  $sp, $sp, -4
  li    $v0, 0                       # hoisted
  nop                                # cannot hoist anything out of loop
                                     # body -- bypassing would eliminate
  sw    $ra, 0($sp)
count_top:
  lb    $t0, 0($a1)
  nop                                # no choice but to wait, addi must
  nop                                # come after both branches fall thru
                                     # bypassing would eliminate one nop
  beq   $t0, $zero, count_done
  bne   $t0, $a0, count_advance
  addi  $v0, $v0, 1
count_advance:
  addi  $a1, $a1, 1
  nop                               # bypassing would eliminate (addi to lb at count_top
  j     count_top
count_done:
  lw    $ra, 0($sp)
  addi  $sp, $sp, 4
  nop                               # no useful work left to put here --
                                    # bypassing would eliminate
  jr    $ra
```

3. (25 pts.) Suppose the MIPS processor were divided into 10 stages, s1 through s10. The key operations are distributed across the stages as follows:

s1	**s2**	**s3**	**s4**	**s5**	**s6**	**s7**	**s8**	**s9**	**s10**
	branch resolution		register decode	ALU			data memory		register writeback

Hint: You can ignore conditional branches for this problem.

(a) What is the ideal CPI of this pipeline?

1 cycle per instruction

(b) Assuming no operand fordwarding, what is the CPI of producer and consumer instructions?

Without forwarding, operands must be passed through the register file, written in S10 and read in s5. Such dependent instructions can be processed at a rate of one instruction every six cycles, for a CPI of 6.

(c) Consider an instruction mix of 25% loads, 10% stores, 2% jumps, and 52% R-type instructions. Assuming that 50% of the loads and R-type instructions are followed immediately by their consumers, and that there is no data forwarding, what is the average CPI of this workload?

The CPI will be 1 for all of the instructions except for the half of loads (12.5%) and R-types (26%) that have a CPI of 6. So, the overall CPI is $6 \times 0.385 + 1 \times (1 - 0.385) = 2.925$.

(d) What is the average CPI on the standard 5 stage pipeline, also without operand forwarding?

The CPI remains 1 for everything except the same bubble-requiring instructions. These require two bubbles, resulting in a CPI of 3 for those instructions. So the overall CPI is $3 \times 0.385 + 1 \times (1 - 0.385) = 1.77$

(e) How much faster does the 10-stage pipe's clock need to be than the 5-stage in order for the two processors to have the same performance on this workload?

$$\begin{aligned}
CPI_{10} \times \frac{1}{f_{10}} &= CPI_5 \times \frac{1}{f_5} \\
2.925 \times \frac{1}{f_{10}} &= 1.77 \times \frac{1}{f_5} \\
2.925 \times f_5 &= 1.77 \times f_{10} \\
\frac{2.925}{1.77} &= \frac{f_{10}}{f_5} \\
&= 1.65 \text{ times faster}
\end{aligned}$$

4. (25 pts.) Consider the **direct mapped** cache that interprets an 8-bit address according to the `{tag:setIdx:byteOffset}` format specified.

   (a) Complete the table on the following page, indicating whether the each access in the reference stream will hit (H) or miss (M) in each cache.

   (b) If the L1 access time is 7ps, and memory has a 100ps access time, what is the expected access time for the references above on the two level hierarchy?

   $EAT = AT_{L1} + MR_{L1} \times AT_M = 7 + 0.66 \times 100 = 73ps$

   (c) Will the addition of an L2 cache with a 50% miss rate and 20ps access time improve or degrade the hierarchy?

   The time to resolve an L1 miss is now $20 + 0.5 \times 100$ (to access the L2 and memory) which is less than the 100 it was before, so this change will improve the hierarchy.

Address Format →	**L1 {2:2:4}**
Block Size (Bytes):	16
Cache Size (Blocks):	4

Address References (in binary)

01. 00. 0000	M
00. 10. 0000	M
00. 01. 0000	M
00. 00. 1000	M
00. 00. 0100	H
00. 00. 0010	H