CSEE W3827
Fundamentals of Computer Systems
Homework Assignment 6
Solutions

Prof. Martha A. Kim
Columbia University
Due December 12, 2013 at 12 noon, in 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. (20 pts.) Consider the execution of the program below on a fully bypassed (i.e., both W-E and M-E operand forwarding), 5-stage MIPS pipeline with early branch resolution (i.e., branches resolved in D stage).

```
i1:    add     $v0, $0, $0
i2:    addi    $t0, $0, 100
i3:    lw      $t1, 0($a0)
i4:    add     $v0, $v0, $t1
i5:    addi    $a0, $a0, 4
i6:    addi    $t0, $t0, -1
i7:    beq     $t0, $0, end
i8:    beq     $0, $0, i3
end:
```
(a) Complete the list of operands that are forward when the instructions below are executed, e.g.,

- \( t1 \rightarrow i4 \) (requires stall)
- \( t0 \rightarrow i7 \) (requires stall)

(b) Reorder the instructions so that no stalls are required

```
i1:    add   $v0, $0, $0
i2:    addi  $t0, $0, 100
i3:    lw    $t1, 0($a0)
i5:    addi  $a0, $a0, 4
i6:    addi  $t0, $t0, -1
# separates i6 and i7, also delays lw consumer
i4:    add   $v0, $v0, $t1
i7:    beq   $t0, $0, end
i8:    beq   $0, $0, i3
end:
```
2. (20 pts.) Give an example instruction sequence that would not function if each of the three indicated wires were cut. (The wire cuts are not cumulative, but individual.)
(a) (A) Any sequence that requires a bubble, e.g.,

\[
\begin{align*}
&\text{lw } \$t0, 0(\$t1) \\
&\text{add } \$t1, \$t0, \$t0
\end{align*}
\]

(b) (B) Any sequence that requires forwarding M to the first operand of X, e.g.,

\[
\begin{align*}
&\text{add } \$t0, \$t1, \$t1 \\
&\text{add } \$t1, \$t0, \$t1
\end{align*}
\]

(c) (C) Any sequence that requires detection of a hazard on the destination register of the instruction in M, e.g., the sequence above will also fail here.
3. (20 pts.) Below is the program from the previous homework. Compute the CPI of this program on the *pipelined* processors listed. You may ignore the cycles required to fill/drain the pipe.

```
addi  $s0, $0, 0  # i = 0
add   $s1, $0, $0  # sum = 0
addi  $t0, $0, 10 # $t0 - 10

loop:
    slt  $t1, $s0, $t0  # if (i < 10), $t1 == 1, else $t1 = 0
    beq  $t1, $0, done  # if (i >= 10), branch to done
    add  $s1, $s1, $s0  # sum += i
    addi $s0, $s0, 1    # i++
    beq  $0, $0, loop

done:
```

(a) Early branch resolution, but no forwarding, can only stall to resolve hazards.

Producers and consumers must be separated by 2 instrs (W-D).

- Initialization: addi, add, addi. 2 bubbles on entry to main loop on addi-slt. [3 instrs + 2 bubbles = 5 cycles]
- Main loop iteration: slt, beq, add, addi, beq. 2 bubbles required on slt-beq, otherwise all data dependencies OK including backedge. When branch taken, one instruction will be flushed, so 1 lost cycle per taken branch. [10 * (5 instrs + 2 bubbles + 1 flush) = 80 cycles]
- Exit loop iteration: slt, beq. 2 bubbles required, plus branch taken so one lost cycle. [2 instrs + 2 bubbles + 1 flush = 5 cycles]

In all, this comes to 90 cycles and 55 instructions or 1.63 cycles/instr.
(b) Early branch resolution, fully bypassed.

Only lw producers or beq consumers require bubbles.

- Initialization: addi, add, addi. addi-stl on entry to main loop handled bt forward. [3 instrs = 3 cycles]
- Main loop iteration: slt, beq, add, addi, beq. 1 bubble required on slt-beq, otherwise all data dependencies OK including backedge. When branch taken, one instruction will be flushed, so still 1 lose cycle per taken branch. [10 * (5 instrs + 1 bubbles + 1 flush) = 70 cycles]
- Exit loop iteration: slt, beq. 1 bubbles required, plus branch taken so one lost cycle. [2 instrs + 1 bubbles + 1 flush = 4 cycles]

In all, this comes to 77 cycles and 55 instructions or 1.4 cycles/instr.
4. (20 pts.) Consider a 1024B L1 direct-mapped cache with 8B lines.

(a) For the following ten references, indicate whether they will hit or miss in the cache (assuming it starts empty). (3 bit byte offset, 7 bit set index)

- 0x0000 set 0 → miss
- 0x0008 set 1 → miss
- 0x0000 set 0 → hit
- 0x0010 set 2 → miss
- 0x0002 set 0 → hit
- 0x0012 set 2 → hit
- 0x0080 set 16 → miss
- 0x0000 set 0 → hit
- 0x0800 set 0 → miss
- 0x0880 set 16 → miss

(b) Assuming this cache has an access time of 5 cycles and is backed by memory with 100 cycle access time, what miss rate is required so that the overall access time of the L1-Memory hierarchy is less than 10 cycles?

\[ 10 \text{ cycles} = 5 \text{ cycles} + (R \times 100 \text{ cycles}) \]

The miss rate must be less than 5%
5. (20 pts.) For each of the four caches listed below, break the 32-bit addresses into *tag*, *set index*, and *byte offset* fields.

(a) 1024B capacity, direct mapped, 256B lines

\[
\begin{array}{c}
\text{tag (22 bits)} \quad \text{set index (2 bits)} \quad \text{byte offset (8 bits)}
\end{array}
\]

(b) 4096B capacity, direct mapped, 64B lines

\[
\begin{array}{c}
\text{tag (20 bits)} \quad \text{set index (6 bits)} \quad \text{byte offset (6 bits)}
\end{array}
\]

(c) 1024B capacity, 4-way, 128B lines

\[
\begin{array}{c}
\text{tag (24 bits)} \quad \text{set index (1 bit)} \quad \text{byte offset (7 bits)}
\end{array}
\]