ICS 233, Term 172

Computer Architecture & Assembly Language

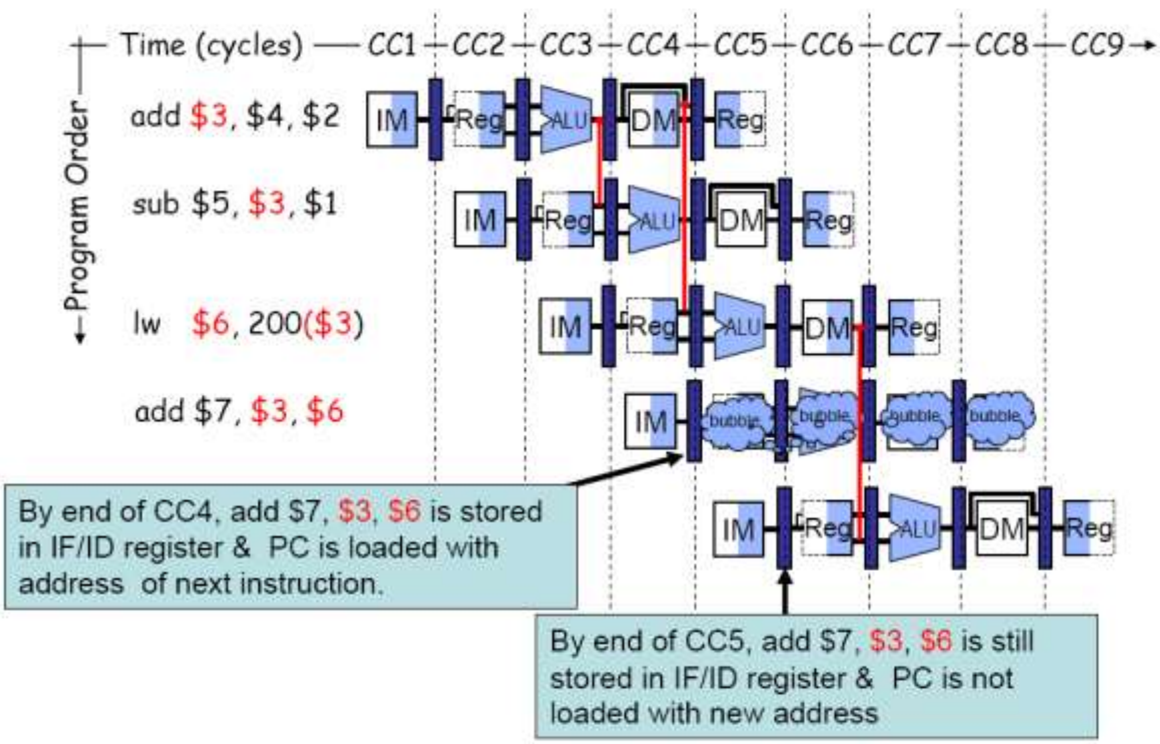
HW# 7 Solution

Q.1. Identify all of the RAW data dependencies in the following code. Which dependencies are data hazards that will be resolved by forwarding? Which dependencies are data hazards that will cause a stall? Using a multiple-clock-cycle graphical representation, show the forwarding paths and stalled cycles if any.

```
add $3, $4, $2
sub $5, $3, $1
lw $6, 200($3)
add $7, $3, $6
```

RAW dependencies:

```
add $3, $4, $2    and    sub $5, $3, $1 (forwarding)
add $3, $4, $2    and    lw $6, 200($3) (forwarding)
lw $6, 200($3)    and    add $7, $3, $6 (stall 1 cycle)
add $3, $4, $2    and    add $7, $3, $6 (from register)
```



Q.2. We have a program of 10^6 instructions in the format of "**lw,add,lw,add,…**". The **add** instruction depends only on the **lw** instruction right before it. The **lw** instruction also depends only on the **add** instruction right before it. If this program is executed on the 5-stage MIPS pipeline:

(i) Without forwarding, what would be the actual CPI?

Without forwarding, the value being written into a register can only be read in the same cycle. As a result, there will be a bubble of 3 cycles between a LW and the dependent ADD to allow the LW to progress through the MEM and WB stages. Similarly, there will be a bubble of 3 cycles between an ADD and the dependent LW. Therefore, it takes 8 cycles on average to complete one LW and one ADD.
1 cycle (to complete LW) + 3 cycles (bubbles) + 1 cycle (to complete ADD) + 3 cycles (bubbles) = 8 cycles
So, it takes 8 cycles to complete 2 instructions
Average CPI = 8/2 = 4.

(ii) With forwarding, what would be the actual CPI?

With forwarding, there will be a bubble of 1 cycle between a LW and the dependent ADD. However, no bubble exists between an ADD and the dependent LW. Therefore, it takes only 3 cycles on average to to complete one LW and one ADD.
1 cycle (to complete LW) + 1 cycle (bubble) + 1 cycle (to complete ADD) = 3 cycles
So, it takes 3 cycles to complete 2 instructions
Average CPI = 3/2 = 1.5.

Q.3. Suppose a computer's address is k bits (using byte addressing), the cache data size is S bytes, the block size is $B = 2^b$ bytes, and the cache is m-way set-associative. Figure out what the following quantities are in terms of S, B, m, b, and k.

(i) The number of sets in the cache.

Number of sets = $S / (B \times m)$

(ii) The number of index bits in the address

Number of index bits in address = $\log_2 (S / (B \times m)) = \log_2(S) - \log_2(B) - \log_2(m)$
$= \log_2(S) - b - \log_2(m)$

(iii) The number of tag bits in the address

Number of tag bits in address = $k - b - \log_2(S) + b + \log_2(m)$
$= k - \log_2(S) + \log_2(m)$

(iv) The total number of bits required to store all the valid and tag bits in the cache

Total number of valid and tag bits in cache =
$m \times (1 + k - \log_2(S) + \log_2(m)) \times S / (B \times m) =$
$(1 + k - \log_2(S) + \log_2(m)) \times S / B$

Q.4. Consider a processor with a 2 ns clock cycle, a miss penalty of 20 clock cycles, a miss rate of 0.05 misses per instruction, and a cache access time (hit time) of 1 clock cycle. Assume that the read and write miss penalties are the same.

(i) Find the average memory access time (AMAT).

AMAT = Hit time + Miss rate × Miss penalty
= 2 ns + 0.05 × (20 × 2 ns) = 4 ns

(ii) Suppose we can improve the miss rate to 0.03 misses per instruction by doubling the cache size. However, this causes the cache access time to increase to 1.2 cycles. Using the AMAT as a metric, determine if this is a good trade-off.

AMAT = 1.2 × 2 ns + 0.03 × 20 × 2 ns = 2.4 ns + 1.2 ns = 3.6 ns
Yes, this is a good trade-off.

(iii) If the cache access time determines the processor's clock cycle time, which is often the case, AMAT may not correctly indicate whether one cache organization is better than another. If the processor's clock cycle time must be changed to match that of a cache, is this a good trade-off? Assume that the processors in part **(i)** and **(ii)** are identical, except for the clock rate and the cache miss rate. Assume 1.5 references per instruction (for both I-cache and D-cache) and a CPI without cache misses of 2. The miss penalty is 20 cycles for both processors.

CPU time = Clock cycle × IC × ($CPI_{ideal\text{-}cache}$ + cache stall cycles per instruction)
CPU time(i) = 2 ns × IC × (2 + 1.5 × 20 × 0.05) = 7 × IC
CPU time(ii) = 2.4 ns × IC × (2 + 1.5 × 20 × 0.03) = 6.96 × IC
The CPU times in parts (i) and (ii) are almost identical. Hence, doubling the cache size to improve the miss rate at the expense of stretching the clock cycle, results in essentially no net gain.