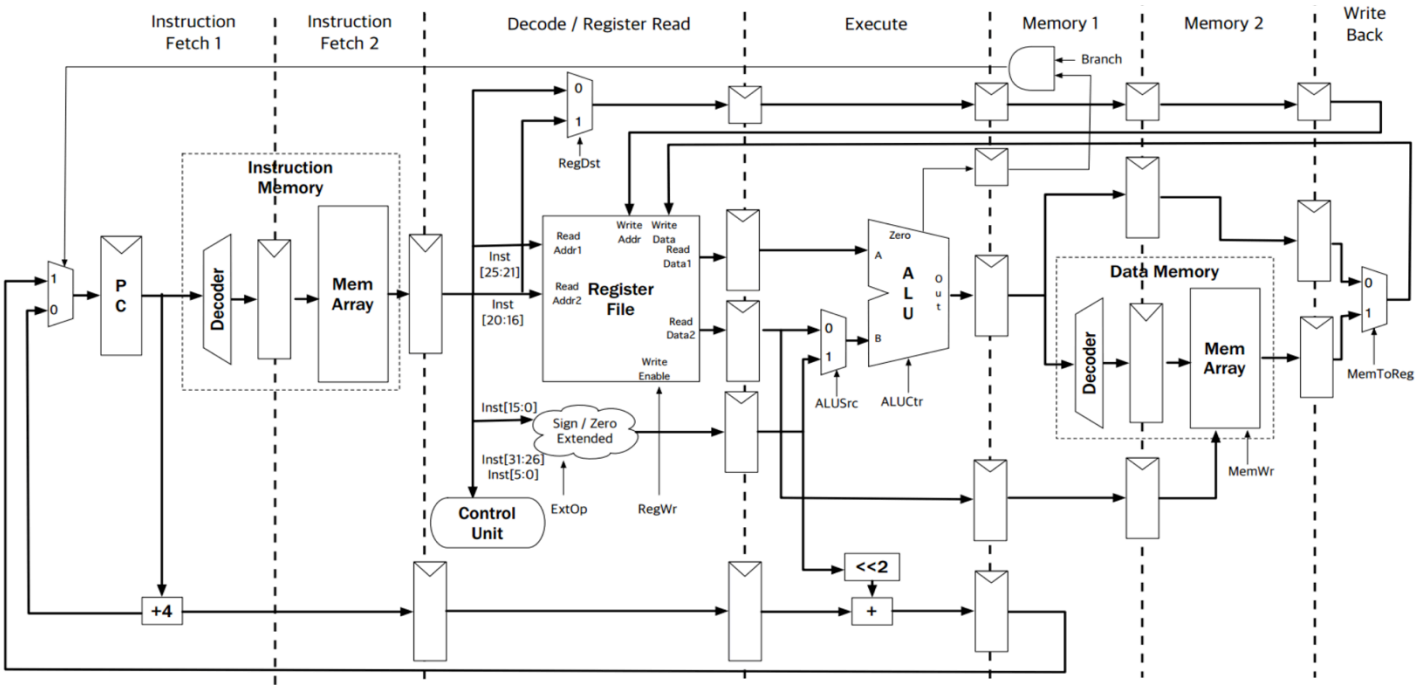
CS61C Summer 2015
Guerrilla Session 4: Pipeline Hazards & Caches

1) If this exam were a CPU, you'd be halfway through the pipeline (Sp15 Final)

We found that the instruction fetch and memory stages are the critical path of our 5-stage pipelined MIPS CPU. Therefore, we changed the IF and MEM stages to take two cycles while increasing the clock rate. You can assume that the register file is written at the falling edge of the clock.



Assume that no pipelining optimizations have been made, and that branch comparisons are made by the ALU. Here's how our pipeline looks when executing two add instructions:

Clock Cycle #	1	2	3	4	5	6	7	8
add $t0, $t1, $t2	IF1	IF2	ID	EX	MEM1	MEM2	WB	
add $t3, $t4, $t5		IF1	IF2	ID	EX	MEM1	MEM2	WB

Make sure you take a careful look at the above diagram before answering the following questions:

a) How many stalls would a data hazard between back-to-back instructions require?

3 stalls

b) How many stalls would be needed after a branch instruction?

4 stalls

c) Suppose the old clock period was 150 ns and the new clock period is now 100ns. Would our processor have a significant speedup executing a large chunk of code...

i) Without any pipelining hazards? Explain your answer in 1-2 sentences.

Yes, due to 1.5x throughput

ii) With 50% of the code containing back-to-back data hazards? Explain your answer in 1- 2 sentences.

Yes, penalty is 300 ns per hazard in both cases, so our new processor will still have higher throughtput.

2) Watch Your (Time) Step! (Su12 Final)

Consider the following difference equation (i.e. differential equation that is discretized in time):

```
x[n+1] - x[n] = -alpha*x[n]
x[n+1] = (1-alpha)*x[n]
```

Here we restrict ourselves to integer values of alpha and use the following integration function:

```
        # $a0 -> addr of array to store x[i] (assume x[0] is already set)
        # $a1 -> length of array/integration
        # $a2 -> alpha
      integrate:
1           beq   $a1,$0,exit
2           sub   $t0,$0,$a2        # $t0 = -alpha
3           addi  $t0,$t0,1         # $t0 = 1-alpha
4           lw    $t1,0($a0)        #$t1=x[n]
5           mult  $t0,$t1
6           mflo  $t1               # $t1 = (1-alpha)*x[n]
7           sw    $t1,4($a0)        # x[n+1] = (1-alpha)*x[n]
8           addiu $a0,$a0,4
9           addiu $a1,$a1,-1
10          j     integrate
11    exit: jr    $ra
```

We are using a 5-stage MIPS pipelined datapath with separate I$ and D$ that can read and write to registers in a single cycle. Assume no other optimizations (no forwarding, etc.). The default behavior is to stall when necessary. Multiplication and branch checking are done during EX and the HI and LO registers are read during ID and written during WB.

a) As a reminder, Tc stands for "time between completions of instructions." Given the following datapath stage times, what is the ratio Tc,single-cycle/Tc,pipelined?

IF	ID	EX	MEM	WB
200 ps	100 ps	400 ps	200 ps	100 ps

1000/400=5/2

b) Count the number of the different types of potential hazards found in the code above:

Structural:_____0_____ Data:_____6_____ Control:_____3_____

None @ 2-3, 3-5, 4-5, 5-6, 6-7, 9-1 @ 1, 10, 11

For the following questions, examine a SINGLE ITERATION of the loop (do not consider the jr).

c) With no optimizations, how many clock cycles does our 5-stage pipelined datapath take in one loop iteration (end your count exactly on completion of j)?

24 = 10 instructions + 4 for draining pipeline + 2*(# data hazards – 2) + 2 for beq control hazard. Beq hazard is 2 stalls because branch comparison done in EX stage.

data hazards minus 2 because 3-5 and 9-1 hazards taken care of by other stalls.

d) How many clock cycles LESS would be taken if we had FORWARDING?

7= 2*(# of data hazards – 1) – 1 because of load hazard. Again, you can ignore the 3-5 hazard here.

e) Assuming that we introduce both FORWARDING and DELAY SLOTS, describe independent changes to integrate that will reduce the total clock cycles taken. Changes include replacing or moving instructions. You may not need all spaces given.

Instr	**Change**
___3____	Move after 4 (or move instr 4 before instr 3)_____
___8/9__	Move after 10_____

OR

____9___	Move after 4_____
____8___	Move after 10_____

3) Cache, gotta hit it! Missing it would be painful (Su14 Final)

(a) Following statements are related to cache optimization. Specify whether each of the following statements would generally be true or false.

T / F – Assuming the same cache size and the same block size, increasing set associativity of a cache reduces conflict misses.

T / F – Assuming the same set associativity and the same block size, increasing the size of a cache reduces compulsory misses.

T / F – Smaller caches have shorter hit time than larger caches.

T / F – Adding a lower level cache reduces miss penalty.

T / F – Adding a lower level cache increases hit time.

T / F – Increasing set associativity increases hit time.

For following questions through (b) to (d), we will be working on a 32-bit byte-addressed MIPS machine, with a single direct mapped 1KiB cache, write-through and no-write allocate policies, and 16B blocks. The cache is initially cold for each question.

Consider the function `matrix_multiply` listed below, which multiplies the matrix A by itself and stores the result into the matrix C. Note that starting addresses of each Matrix are as in comments.

```
#define N 4
int A[N * N]; // starts at address 0x10000
int C[N * N]; // starts at address 0x20000
void matrix_multiply() {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
               C[i*N+j] += A[i*N+k] * A[k*N+j];
```

```
            }
        }
    }
}
```

(b) What is the cache hit rate of the function `matrix_multiply` while i = 0?
13/48

(c) If our cache is two-way set-associative with LRU replacement policy, what would be the hit rate of the function `matrix_multiply` while i = 0?
43/48

(d) If our cache is two-way set-associative with LRU replacement policy and its block size is 32B, what would be the hit rate of the function `matrix_multiply` while i = 0?
15/16 (= 45/48)

4) Cache, money y'all (Sp14 Final)

We have a standard 32-bit byte-addressed MIPS machine with 4 GiB RAM, a 4-way set-associative CPU data cache that uses 32 byte blocks, a LRU replacement policy, and has a total capacity of 16 KiB. Consider the following C code and answer the questions below.

```
#define SIZE_OF_A 2048
typedef struct {
    int x;
    int y[3];
} node;
int count_x(node *A, int x) {
    int k = 0;
    for (int i = 0; i < SIZE_OF_A; i++)
        if (A[i].x == x) {
            k++;
        }
    return k;
}
```

a) How many bits are used for the tag, index, and offset? 20:7:5

b) We call count_x with all values of x from 0 to 65535 to count the number of times that each x occurs in A. The value of A is the same in every call. The cache is cold at the beginning of execution. What is the cache hit rate? 50%

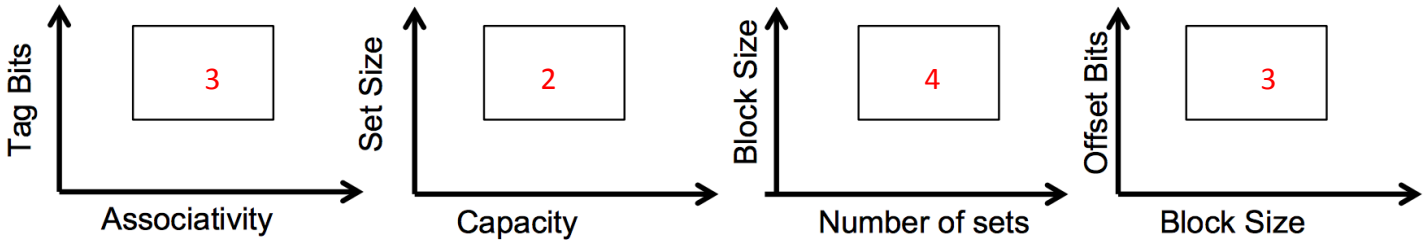
Questions (c) and (d) below are two independent variations on the original problem.

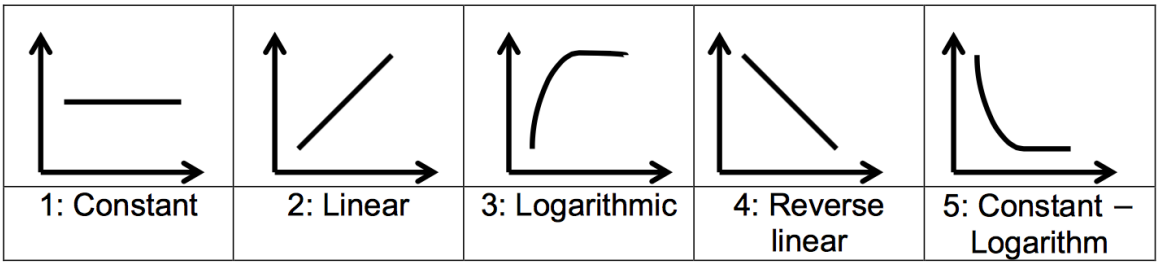
c) Let's say that we increase our CPU cache associativity to 8-way. What is our cache hit rate now? 50%

d) What would be the approximate cache rate if we changed our CPU cache to use a Most Recently Used (MRU) cache replacement policy, and we change the cache to be fully associative? 75%
Half the array fits in cache, on the first cold pass it'll be (first-half-minus-last-block, last-block) at 50%. The next iteration will hit all of the first-half-minus-last-block (100% hits), then the "last block in the first half" will kick out the "all but" last block, but then you get a hit on the last block too, so it's exactly half with 100% hits, and half with 50% hits.

5) Some say there's nothing better than cold, hard cache (Sp15 Final)

a) What shape do the following trade-off curves have? Select a shape and enter its number into the box for each of the graphs. Unless they are the parameters being varied, assume that associativity, capacity and block size are constant. You should assume that the axes are linear.





b) Consider a system with inclusive L1 and L2 caches with 4B cache block size. Assume we have 1 MiB of on-chip memory available and want to determine how much of this memory we should give to the L1 cache and how much to the L2 cache. We will try to minimize the AMAT to do so.

Assume both caches are fully associative with LRU replacement. Their combined capacity is 1MiB (excluding tags and meta-data). **You can consider all miss rates approximate.**

Say you are running the following program starting from cold L1 and L2 caches:

```
#define ARRAY_SIZE 256*1024
int a[ARRAY_SIZE];
int sum = 0; // assume sum, i, and j are stored in registers
for (int i = 0; i < 100000; i++) {
  for (int j = 0; j < ARRAY_SIZE; j++) sum += a[j];
  for (int j = ARRAY_SIZE-1; j >= 0; j--) sum += a[j];
}
```

1) How would we compute AMAT if we had the local L1 miss rate (“L1Miss”), the local L2 miss rate (“L2Miss”) and the memory access time (“Memory”)? Use “H1” and “H2” to represent the L1 and L2 hit times respectively. (We will compute these quantities later in the question)
AMAT = H1 + L1Miss * (H2 + L2Miss * Memory)

2) For the program above, express the local miss rate for the L1 cache in general terms as a function of the L1 cache size (write L1 for the size of L1 in bytes). Hint: The miss rate is 0 for a 1 MiB cache, 0.5 for a 0.5 MiB cache and 1 for a 0 MiB (i.e., no) L1 cache.
LocalMiss1 = 1 – (L1/1MiB)

3) What is the global miss rate for the L2 cache as a function of the L1 cache size? Hint: Start by expressing the global miss rate as a function of the L2 cache size.
GlobalMiss2 = 1-(L2/1MiB) = (1MiB-L2)/1MiB = L1/1MiB

4) What is the local miss rate for the L2 cache as function of the L1 and L2 sizes? Hint: Use your results from questions 2 and 3.
LocalMiss2 = [L1/1MiB] / [1 – (L1/1MiB)] = L1/(1MiB-L1) = L1/L2

5) Assume the hit time of the L1 cache is 10 cycles, the hit time of the L2 cache is 20 cycles and the memory access time is 100 cycles. Using the formula from question 1, what is the AMAT for this system as a function of only the L1 size?
AMAT = H1 + LocalMiss1 * (H2 + LocalMiss2 * Memory) = 10 + [L2/1MiB]*[20 + (L1/L2)*100]
= 10 + (1MiB-L1)*20/1MiB + L1*100/1MiB
= 10 + 20 – 20*L1/1MiB + 100*L1/1MiB
= 30 + 80*L1/1MiB

6) What sizes of L1 and L2 caches should we pick to minimize the AMAT? (assume the caches have non-zero size, i.e., both of them exist)
L1 = 4B, L2 = 1MiB-4B