CSCI 6461: Computer System Architecture – Homework 5

Shubham Jadhav | G30570862

3.18)

To find: How much faster is the processor with the branch-target buffer versus a processor that has a fixed two-cycle branch penalty

Given:

Miss prediction penalty: **4 cycles**
Buffer miss penalty: **3 cycles**
Hit rate: **90%**
Prediction accuracy: **90%**
Branch frequency: **15%**
Base CPI w/o branch stall: **1**
Processor branch penalty: **2**

We can calculate, the performance comparison as follows:

Speedup = $\frac{(Clock\ per\ instructions\ without\ branch\ target\ buffer)}{(Clock\ per\ instructions\ with\ branch\ target\ buffer)}$

CPI w/o BTB = Base CPI + Base Stalls
= Base CPI + (Branch Frequency x Processor branch penalty)
= 1 + (0.15 x 2)
= **1.3**

CPI w BTB = Base CPI + BTB Stalls

Stalls of BTB depends on various events

If BTB result is miss:
Stall = Base Frequency x miss rate x miss penalty
= 0.15 x 0.1 x 3
= 0.045

If BTB result is hit with correct prediction, then stall will be 0.

If BTB result is hit with incorrect prediction:
Stall = Base Frequency x hit rate x prediction failure x miss prediction penalty
= 0.15 x 0.9 x (1 – prediction accuracy) x 4
= 0.15 x 0.9 x 0.1 x 4
= 0.054

Total Stalls = 0.045 + 0.054 = 0.097

CPI w BTB = Base CPI + BTB Stalls
= 1 + 0.097
= **1.097**

Therefore, Speedup = $\frac{1.3}{1.097}$ = 1.185 ≈ **1.2**

4.9)

Given:

Processor frequency: 700 MHz
Vector length: 64 bits i.e., 8 bytes
Startup Overheads:

- Load/Store: 15 cycles
- Multiple unit: 8 cycles
- Add/subtract unit: 5 cycles

a) Arithmetic intensity is a measure of floating-point operations performed by a given code relative to the amount of memory accesses that are required to support those operations. It is frequently described as a FLOP per Byte ratio (F/B).

Arithmetic intensity = $\frac{floating-point\ operations}{data\ bytes\ transfered}$

In the given code, we can see that there are 4 floating read operations and 2 floating write operations for every 6 bytes transferred.

Therefore, Arithmetic intensity = $\frac{4+2}{6} = \frac{6}{6} = \mathbf{1.}$

c)

vmul	vld	a_re *x* b_re , load a_im	#1
vld	vmul	load b_im , a_im *x* b_im	#2
vsub	vst	subtract and store c_re	#3
vmul	vld	a_re *x* b_im , lead next a_re vector	#4
vmul	vld	a_im *x* b_re , load next a_re	#5
vadd	vst	add and store c_im	#6

Total of **6** chimes are required for chaining and a single memory pipeline.

d)

To find: clock cycles per complex result value with overhead for chained vector sequence

For total cycles
= [total chimes x elements] + [Load/Store overhead x 6] + [multiple overhead x 4] + [add/subtract overhead x 2]
= [6 x 64] + [15 x 6] + [8 x 4] + [5 x 2]
= **516 cycles**

For total cycles per result = $\frac{516}{128}$ = **4 cycles**

e)

vmul	a_re *x* b_re	#1
vmul	a_im *x* b_im	#2
vsub vst	subtract and store c_re	#3
vmul	a_re *x* b_im	#4
vmul vld	a_im *x* b_re , load next a_re	#5
addvv.s vst vld vld vld	add , and store c_im , load next b_re , a_im , b_im	#6

Total of **6** chimes are required for chaining and a three memory pipeline. There is no performance improvement as compared to single memory pipeline, even after adding additional load/store units.