QUIZ 5 SOLUTIONS
CS152 – Spring 2008

Krste Asanovic

May 6, 2008

Problem Q5.1	How is vertical waste caused by long latency instructions reduced?	How is horizontal waste caused by wide issue widths reduced?	Where and by what is most parallelism extracted (e.g., programmer, compiler, hardware)?	Name a limitation or disadvantage as compared to a simple in-order execution RISC machine
Out-of-Order Execution				
VLIW				
Vector				
Multithreading				
Simultaneous Multithreading				

Problem Q5.2: Predication and VLIW 15 Points

Problem M5.2.A

```
        l.s    f1, 0(r1)      ; f1 = *r1
        seq.s r5, f10, f1     ; r5 = (f10==f1)
        cmpnez p1, r5         ; p1 = (r5!=0)
(p1)    add.s f2, f1, f11     ; if (p1) f2 = f1+f11
(!p1)   add.s f2, f1, f12     ; if(!p1) f2 = f1+f12
        s.s    f2, 0(r2)      ; *r2 = f2
```

Problem Q5.3: Multithreaded architectures

Problem Q5.3.A

4, largest latency for any instruction is 4

Problem Q5.3.B

2/12 = 0.17 flops/cycle (two flops per loop, on average we complete a loop every 12 cycles)

Problem Q5.3.C

Yes, we can hide the latency of the floating point instructions by moving the add instructions in between floating point and store instructions – we'd only need 3 threads. Moving the third load up to follow the second load would further reduce thread requirement to only 2.

Problem Q5.4: Vectorization 15 Points

State whether each of the following loops could be successfully vectorized and explain your answer. In all cases, you should assume that arrays **A**, **B**, **C** do not overlap in memory.

`for (i=0; i<N; i++)` `    B[i] = A[i] + C;`	Yes. C was supposed to be considered a scalar value. Scalars can be added to vectors by adding the value to each element of the vector. These additions are all independent.
`for (i=1; i<N; i++)` `    B[i] = A[i] + B[i-1];`	No. To vectorize this, we would have to create vectors out of arrays A and B, and then operate on all elements in parallel. However, the value assigned to each element of B is dependent on the value assigned to the previous elements, i.e. there is a dependency. Chaining does not solve this problem because it works only between consecutive vector instructions, not between the elements of a single vector instruction.
`for (i=0; i<N-1; i++)` `    B[i] = A[i] + B[i+1];`	Yes. Could be done by creating a vector from the elements of B from 1 to N, adding this to a vector created from the elements of A from 0 to N-1, and then writing back the result. In other words, reads and writes to B do not have to be interleaved so there are no dependencies limiting vectorization.
`for (i=0; i<N; i++)` `    C[i] = A[B[i]];`	Yes, by using a gather operations. See lecture slides.

`for (i=0; i<N; i++)`
`    if(C[i] != 0)`
`        B[i] = A[i] + D;`
	Yes, by using flags/vector masks. See lecture slides.