263-2300-00: How To Write Fast Numerical Code

Assignment 2: 80 points

Submitted by Jinank Jain

Solution 1

Solution 2

Part b:
In this part I tried four different variations which are discussed as below:

- In the first approach I just unroll the inner loop and does not do any modifications with that unrolled loop. The function is called `loop_unroll` in C file.
- In the second approach, I did some optimization with unrolled loop that I did in the previous section, I replaced $x[i]$ with some some scalar a which stores the value of $x[i]$ in some register. Another optimization was to much multiple accumulator instead single sum. I used for different sum variable to store the intermediate result. The function is called `scalar_replacement` in C file.
- In the third approach I unrolled the first loop too. Since in the array y there are only 5 values and I want them to be stored in registers so I unrolled first loop in multiple of 5 and storing all the values in the register. And creating 20 different sum accumulator and summing up the final answer. The function is called `super_scalar_replacement` in C file.
- In the fourth approach I remove the assumption which I took in third approach that n was divisible by 5 so for that I need introduce some branching instructions. The function is called `super_scalar_replacement_generalize` in C file.

Part d:

Optimization Flags	**slow performance1**	**loop _ unroll**	**scalar _replacement**	**super _scalar _replacement**	**super _scalar _replacement _generalize**
-O0	0.164 FLOPs/c	0.238 FLOPs/c	0.233 FLOPs/c	0.817 FLOPs/c	0.738 FLOPs/c
-O3	0.537 FLOPs/c	0.549 FLOPs/c	0.544 FLOPs/c	4.881 FLOPs/c	3.527 FLOPs/c

Table 1: Performance comparison between different optimization function

Solution 3

Table 2: My caption

Instruction	Reference Values			Regular Case			Special Case: Mul: 1			Special Cas Div: 2	
	Latency	TPS	Gap	Latency	TPS	Gap	Latency	TPS	Gap	Latency	TPS
MULSS	5	1	1	5.34	0.76	1.32					
MULSD	5	1	1	5.06	0.91	1.09					
DIVSS	11-14	0.16	6	13.07	0.13	7.49					
DIVSD	15-20	0.07	14	20.06	0.07	14.06					