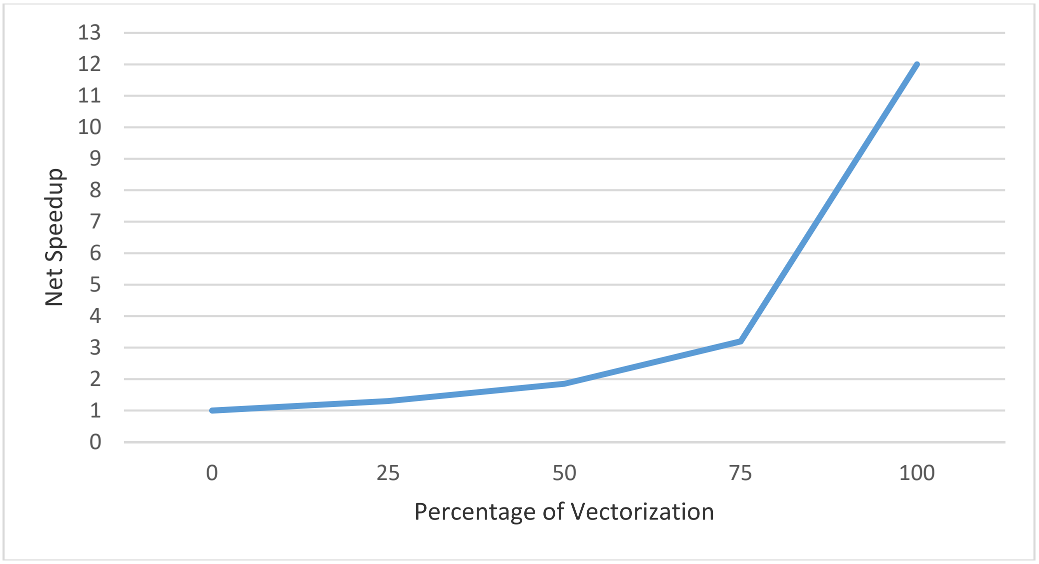
Homework 3 solutions

Questions 2 (Amdahl Law) 6 points: In this exercise, we are considering enhancing a machine by adding vector hardware to it. When a computation is run in vector mode on the vector hardware, it is 12 times faster than the normal mode of execution. We call the percentage of time that could be spent using vector mode, the *percentage of vectorization.*

a) Draw a graph that plots the speedup as a function of the percentage of the computation performed in vector mode.
b) What percentage of vectorization is needed to achieve a speedup of 2?
c) What percentage of vectorization is needed to achieve one-half the maximum possible speedup attainable from using vector mode?
d) Suppose you have measured the typical percentage of vectorization to be 70%. The hardware design group estimates that it can speed up the vector hardware to 20 (instead of 12) with significant additional investment. You wonder if the compiler group could increase the percentage of vectorization, instead. What percentage of vectorization would the compiler group need to achieve in order to equal the improvement proposed by the hardware group?

a)

Percentage of Vectorization (PV)	Net Speedup
0	1
25	$\frac{1}{0.75 + 0.25/12} = 1.3$
50	$\frac{1}{0.50 + 0.50/12} = 1.85$
75	$\frac{1}{0.25 + 0.75/12} = 3.2$
100	12



b) $\frac{1}{(1-PV)+\left(\frac{PV}{12}\right)} = 2; \quad \text{PV} = 54.55\%$

c) Maximum speed-up is 12. One-half of maximum speed up is 6.

$$\frac{1}{(1-PV)+\left(\frac{PV}{12}\right)} = 6; \quad \text{PV} = 90.91\%$$

d) 70% vectorization and 20 times faster hardware yield a net speedup of

$$\frac{1}{0.3 + 0.7/20} = 2.985$$

To achieve this net speedup by improving the compiler, we must increase the percentage of vectorization. We have $\frac{1}{(1-PV)+\left(\frac{PV}{20}\right)} = 2.985$

$fraction_{vectorized} = \ 72.54\%$

Questions 3 – six points: Consider the execution of 11 instructions, A1, …, A11 of a thread A, and 8 instructions B1, …, B8 of a second thread, B, on a single 2-issue pipeline, where in each cycle at most one R-type instruction and one M-type (Memory type) instruction can be issued but instructions cannot be issued out of order (similar to the superscalar that you implemented in Project 1). Moreover, due to data dependencies and constrains on forwarding in the architecture, a scheduler (instruction dispatcher) has to observe the following:

A5 should be issued at least 2 cycle after A3 is issued
A6 should be issued at least 2 cycle after A4 is issued
A9 should be issued at least 4 cycle after A8 is issued
B4 should be issued at least 2 cycle after B3 is issued
B5 should be issued at least 2 cycle after B4 is issued
B6 should be issued at least 1 cycle after B5 is issued

Which leads to the following schedules for each thread when no multithreading is used:

cycle	R-type issued	M-type issued
1	A1	A2
2	A3	
3	A4	
4		A5
5	A6	A7
6	A8	
7		
8		
9		
10	A9	
11	A10	A11

cycle	R-type issued	M-type issued
1		B1
2	B3	B2
3		
4		B4
5		
6	B5	
7		B6
8	B8	B7
9		
10		
11		

Show using three similar tables the schedule if

a) Coarse grain multithreading is used (switch threads after an idle cycle or after the executing thread terminates)

b) Fine grained multithreading is used. Consider two variations:
   b1) threads from A and B are issued strictly in alternate cycles
   b2) threads issue instructions in alternate cycles, but the order is changed if no instructions can be issued from one thread while some instructions can be issued from the other.

c) Simultaneous multithreading is used (with priority given to thread A -- that is, if two instructions Ai and Bk, that are of the same type, can be issued in the same cycle, Ai is scheduled)

Coarse grain multithreading		
cycle	R-type	M-type
1	A1	A2
2	A3	
3	A4	
4		A5
5	A6	A7
6	A8	
7		
8		B1
9	B3	B2
10		
11	A9	
12	A10	A11
13		B4
14		
15	B5	
16		B6
17	B8	B7

Fine grain multithreading(b1)		
cycle	R-type	M-type
1	A1	A2
2		B1
3	A3	
4	B3	B2
5	A4	A5
6		B4
7	A6	A7
8	B5	
9	A8	
10		B6
11		
12	B8	B7
13	A9	
14		
15	A10	A11
16		
17		

Simultaneous multithreading		
cycle	R-type	M-type
1	A1	A2
2	A3	B1
3	A4	B2
4	B3	A5
5	A6	A7
6	A8	B4
7		
8	B5	
9		B6
10	A9	B7
11	A10	A11
12	B8	
13		
14		
15		
16		
17		

Fine grain multithreading(b2)		
cycle	R-type	M-type
1	A1	A2
2		B1
3	A3	
4	B3	B2
5	A4	A5
6		B4
7	A6	A7
8	B5	
9	A8	
10		B6
11	B8	B7
12		
13	A9	
14	A10	A11
15		
16		
17		