Courtesy of Mr. Derrick Sines

February 18, 2002
CDA5106

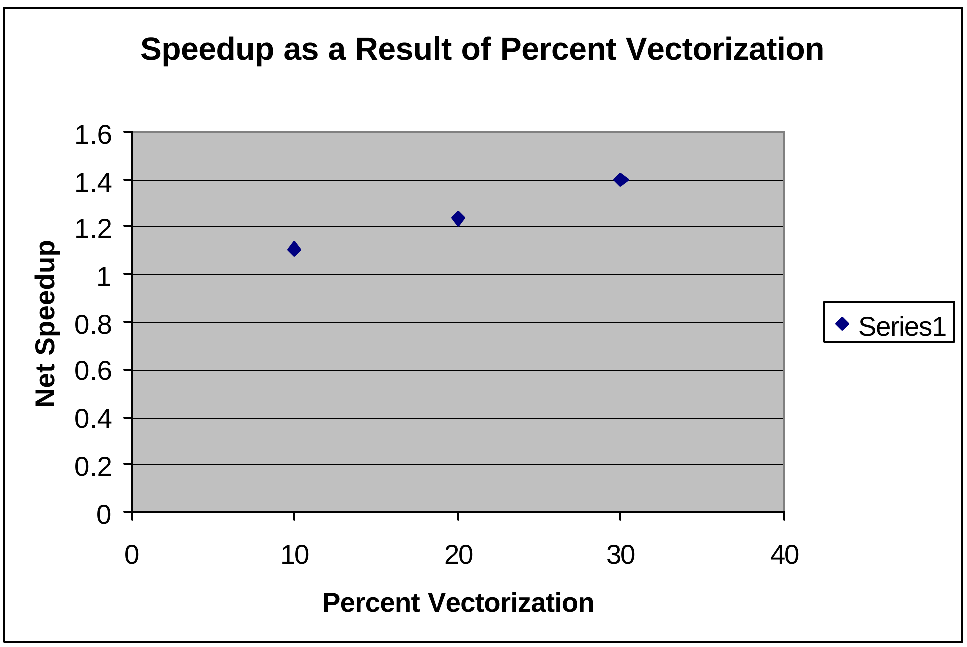
Homework #1
Problems 1.1, 1.2, 1.6, 1.7, 1.8

1.1

a) For the graph the x and y values can be found easily by using the value of 20 for $Speedup_{enhanced}$ and the values 10, 20, and 30 for the percentage of vectorization or $Fraction_{enhanced}$.

Plugging these values into
Net Speedup = $1/((1-Fraction_{enhanced}) + (Fraction_{enhanced}/Speedup_{enhanced}))$
Gives

X	Y
10	1.105
20	1.235
30	1.399



b) Using Amdahl's Law the following equation can be used:
$Speedup_{overall} = 1/((1 – Fraction_{enhanced}) + (Fraction_{enhanced}/Speedup_{enhanced}) = 2$

Rearranging this to solve for $Fraction_{enhanced}$ gives the equation:
$Fraction_{enhanced} = speedup_{overall} * speedup_{enhanced} - speedup_{enhanced}/ speedup_{overall} * speedup_{enhanced} – speedup_{overall}$

So,
$Fraction_{enhanced} =(2 * 20 – 20)/(2 * 20 – 2) = .526$

Therefore, the percentage vectorization needed to gain a speedup of 2 is 52.6%

c)
Since the maximum speedup attainable would be 20, one-half of this value would be a $Fraction_{enhanced}$ = 10.
So,
$Speedup_{overall} = 1/((1 – Fraction_{enhanced}) + (Fraction_{enhanced}/Speedup_{enhanced}) = 10$

Rearranging this to solve for $Fraction_{enhanced}$ gives the equation:
$Fraction_{enhanced} = speedup_{overall} * speedup_{enhanced} - speedup_{enhanced}/ speedup_{overall} * speedup_{enhanced} – speedup_{overall}$

As above,
$Fraction_{enhanced} = (10 * 20 – 20)/(10 * 20 – 10) = .95$

Therefore, the percentage vectorization needed to gain one-half of the maximum speedup attainable is 95%.

d)
Using the hardware doubling the following equation can be generated:
$Speedup_{overall} = 1/((1 – .7) + (.7/40)$
Solving the above equation gives a $Speedup_{overall}$ of 3.149

To find what percentage of vectorization would need to be achieved to match the hardware doubling method simply solve for $Fraction_{enhanced}$ as follows:
$Fraction_{enhanced} = speedup_{overall} * speedup_{enhanced} - speedup_{enhanced}/ speedup_{overall} * speedup_{enhanced} – speedup_{overall}$

So,
$Fraction_{enhanced} = (3.14 * 20 – 20)/(3.14 * 20 – 3.14) = .718$

Therefore, a percentage vectorization of around 1% would match the hardware doubling technique. I would recommend using the compiler to crew to gain the extra 1% vectorization.

1.2

a) Speedup = $Time_{original}/Time_{enhanced}$

Since the speedup value is 10:
$.5Time_{enhanced} = (Fraction_{enhanced} * Time_{orginal})/10$

Solving for $Percentage_{original}$:
$Fraction_{enhanced} = 10 * Time_{enhanced}/2 * Time_{original}$ (Equation 1)

Since 50% of enhanced execution time is spent is used on the enhancement, the following can also be found:

$.5Time_{enhanced} = (1 – Fraction_{enhanced}) * Time_{original}$

Plugging $Fraction_{enhanced}$ found in Equation 1 into the previous equation and solving for $Time_{original}/Time_{enhanced}$ yields:

$Speedup_{overall}$ = 5.5

b) To solve the speedup found in 1.2a must be used with the equation:
$Speedup_{overall} = 1/((1 – Fraction_{enhanced}) + (Fraction_{enhanced}/Speedup_{enhanced})$

Since the $speedup_{overall}$ is 5.5 and the $speedup_{enhanced}$ = 10 there is only one variable left to solve. The following equation can be used:
$5.5 = 1/((1 – Fraction_{enhanced}) + (Fraction_{enhanced}/10)$

Solving for $Fraction_{enhanced}$ the percentage can then be found.

Percentage of original execution time: 91%

1.6

Execution time is the primary concern so the following equation should be used when making the decision:
CPU Time = IC * CPI * Clock Cycle Time

Starting with the un-optimized version we know that CPI is 1 so:
$CPU\ Time_{un}$ = IC * 1 * Clock Cycle Time (Equation 1)

We also know that the clock rate of the un-optimized machine is 5% higher than the optimized machine so:
(100% - 5%)$Clock\ Cycle\ Time_{op}$ = $Clock\ Cycle\ Time_{un}$ (Equation 2)

The optimized machine does not execute 1/3 of the load and store instructions that the un-optimized machine does, and load and store instructions make up 30% of the total instructions so:
$IC_{op} = .9IC_{un}$ (Equation 3)

Using the equations above as well as information stated in the exercise the following equation can be constructed:

$CPU_{op} = .9IC_{un} * 1 * 1.05\ Clock\ Cycle\ Time_{un}$

Now performance of the optimization can be compared with the un-optimized version by using the speedup equation of:

$Speedup_{overall} = (IC_{un} * Clock\ Cycle\ Time_{un})/(.9IC_{un} * 1.05Clock\ Cycle\ Time_{un})$

Solving the previous equation gives a $Speedup_{overall}$ of 1.06. Therefore, by using the optimization technique a 6% increase in performance is realized.

1.7

a) MIPS = Clock Rate/(CPI * 10^6)

$MIPS_{software}$ = (16.67 * 10^6)/(6 * 10^6) = 2.8
$MIPS_{coprocessor}$ = (16.67 * 10^6)/(10 * 10^6) = 1.7

b) Instruction Count = Execution Time * (MIPS * 10^6)

Instruction $Count_{software}$ = 13.6 * (2.8 * 10^6) = 3.8 * 10^7
Instruction $Count_{coprocessor}$ = 1.08 * (1.7 * 10^6) = 1.8 * 10^6

c) Using the coprocessor because each floating point operation definitely corresponds to one instruction find the non floating point instructions:

NFP = (Instruction $Count_{coprocessor}$) - Total Instructions (From Exercise)
So,
NFP = (1.08 * 10^6) – 195, 578 = 1.6 * 10^6

Floating Point Instructions = Total Instructions – NFP
So,
FPI = (3.8 * 10^7) – (1.6 * 10^6) = 3.6 * 10^7

Instructions per floating point operation = Floating Point Instructions/ Floating Point Operations
So,
Instructions per floating point operation = 3.6 * 10^7/195,578 = 185

Therefore, on average, in software 185 integer instructions are required to perform a floating-point operation.

d) MFLOPS = Number of floating point operations in program/(Execution Time * 10^6)
So,
MFLOPS = 195,578/(1.08 * 10^6) = 0.18

1.8

a) To find the number of good dies per wafer use:
Good Dies per Wafer = Dies per Wafer * Die Yield (Equation 1)
Where
Dies per Wafer = $((\pi*(\text{Wafer Diameter}/2)^{2)} / \text{Die area}) - \pi * \text{Wafer Diameter}/(2 * \text{Die Area})^{1/2}$
And
Die Yield = Wafer yield * $(1 + (\text{Defects per unit area} * \text{Die area})/\alpha)^{-\alpha}$

Using a 20 cm wafer, a defect density of 1 cm^2, α of 3, and a wafer yield of 95% the following table can be created by substituting these values into Equation 1 above.

Microprocessor	Dies per Wafer	Die Yield	Good Chips
MIPS 4600	357	.48	171
PowerPC 603	321	.45	144
HP 71x0	128	.21	26
Digital 21064 A	154	.26	40
SuperSPARC/60	94	.15	14

b) Using the good chip total found in 1.8a the die cost would be found by dividing the wafer cost by the total number of good dies per wafer. Doing this yields the die cost as illustrated below.

Microprocessor	Good Chips	Wafer Cost	Die Cost
MIPS 4600	171	$3200	$18.7
PowerPC 603	144	$3400	$23.6
HP 71x0	26	$2800	$107.7
Digital 21064 A	40	$4000	$100
SuperSPARC/60	14	$4000	$285.7

c)

To find the cost for each good, tested, and packaged good use the values found above in part a and b.

Microprocessor	Die Cost	Testing	Packaging	Total Cost
MIPS 4600	$18.7	$1.7	$12	$32.4
PowerPC 603	$23.6	$2	$20	$45.6
HP 71x0	$107.7	$7.9	$70	$185.6
Digital 21064 A	$100	$5.1	$50	$155.1
SuperSPARC/60	$285.7	$6	$30	$321.7

Testing is calculated using:
Testing Cost = (Hourly Cost/Time Unit)/Die Yield

d)
Solving this problem involves doing the same steps as in part a and b above. Instead of using a defect density of 1 cm^2, defect densities of .6 and 1.2 cm^2 can be used. Then step c is repeated in the same fashion using the new data.

Following these steps yields the following:

Defect Density (cm^2)	Packaging	Die Cost	Testing Cost	Total Cost
.6	30	$157.6	$3.3	$190.9
1.2	30	$386.8	$8.1	$424.9

e)

Solving this problem involves following the steps laid out in parts a – c. Instead of using a defect density of 1 cm^2 this problem uses a defect density of .8 cm^2. The steps laid out in parts a – c should be repeated for two α values of 3 and 4.5. Solving the steps laid out above with these new values yield the following:

α	Packaging	Die Cost	Testing Cost	Total Cost
3	$50	$80	$3.9	$133.9
4.5	$50	$83.8	$4.2	$138