Solution for Homework1

Question 1.5

Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.2.

a. Which processor has the highest performance expressed in instructions per second?

(Instruction Count)*CPI=(CPU Time) * (Clock Rate)

	Instruction Count	Instructions/second
P1	$=10*3*10^9/1.5 = 20*10^9$	$2*10^9$
P2	$=10*2.5*10^9/1 = 25*10^9$	$2.5*10^9$
P3	$=10*4*10^9/2.2 = 18.18*10^9$	$1.818*10^9$

P1 has the best performance expressed in instructions per second.

b. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.

Each processor executes a program in 10s.
CPU Time = (Instruction Count) *CPI / (Clock Rate)
(Instruction Count)*CPI=(CPU Time) * (Clock Rate)

	Instruction Count	Cycles = IC *CPI
P1	$=10*3*10^9/1.5 = 20*10^9$	$30*10^9$
P2	$=10*2.5*10^9/1 = 25*10^9$	$25*10^9$
P3	$=10*4*10^9/2.2 = 18.18*10^9$	$40*10^9$

c. We are trying to reduce the time by 30% but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

CPU Time = IC * CPI / Rate

$Rate_{old} = IC * CPI_{old} / (CPU\ Time_{old})$

$Rate_{new} = IC * CPI_{new} / (CPU\ Time_{new})$

$Rate_{new}/Rate_{old} = CPI_{new}/CPI_{old} * (CPU\ Time_{new}/CPU\ Time_{old})$

$Rate_{new}/Rate_{old} = 1.2 * 0.7 = 0.84$

	Instruction Count	CPI_{new}	$Rate_{new}=IC*CPI_{new}/7seconds$
P1	$=10*3*10^9/1.5 = 20*10^9$	1.2*1.5=1.8	5.14GHz
P2	$=10*2.5*10^9/1 = 25*10^9$	1.2*1=1.2	4.28GHz
P3	$=10*4*10^9/2.2 = 18.18*10^9$	1.2*2.2=2.6	6.75GHz

Question 1.6

Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (class A, B, C, and D). P1 with a clock rate of 2.5 GHz and CPIs of 1, 2, 3, and 3, and P2 with a clock rate of 3 GHz and CPIs of 2, 2, 2, and 2.Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D, which implementation is faster?

a. What is the global CPI for each implementation?

Time: No. instr * CPI/ clockrate

Total time P1= $(10^5 + 2 * 10^5 * 2 + 5 * 10^5 * 3 + 2 * 10^5 * 3)/(2.5 * 10^{9}) = 10.4 * 10^{-4}$

Total time P2= $(10^6 * 2 + 2 * 10^5 * 2 + 5 * 10^6 * 2 + 2)/(3 * 10^9) = 6.66 * 10^{-4}$

Let P1 and P2 be CPU_A and CPU_B

Avg CPI for CPU_A = $10.4 * 10^{-4} * 2.5 * 10^9 / 10^6$ = 2.6

Avg CPI for CPU_B = $6.66 * 10^{-4} * 3 * 10^9 / 10^6$ = 2

b. Find the clock cycles required in both cases.

Let P1 and P2 be CPU_A and CPU_B

Clock Cycles A = IC * Avg CPI = $2.6 * 10^6$

Clock Cycles B = IC * Avg CPI = $2 * 10^6$

Thus CPU_B is faster than CPU_A

Question 1.7

Compilers can have a profound impact on the performance of an application. Assume that for a program, compiler A results in a dynamic instruction count of 1.0E9 and has an execution time of 1.1 s, while compiler B results in a dynamic instruction count of 1.2E9 and an execution time of 1.5 s.

a. Find the average CPI for each program given that the processor has a clock cycle time of 1 ns.

CPU Time = IC * CPI / Rate

Rate = 1GHz = $1 * 10^9$ Hz

$1.1 = 10^9 * CPI_A / 10^9$

$CPI_A = 1.1$

$1.5 = 1.2 * 10^9 * CPI_B / 10^9$

$CPI_B = 1.25$

b. Assume the compiled programs run on two different processors. If the execution times on the two processors are the same, how much faster is the clock of the processor running compiler A's code versus the clock of the processor running compiler B's code?

Rate = IC * CPI / (CPU Time)

$Rate_B / Rate_A = (1.2/1) * (1.25/1.1)$

$Rate_B / Rate_A = 1.2 * 1.136 = 1.36$

RateA is slower than RateB

c. A new compiler is developed that uses only 6.0E8 instructions and has an average CPI of 1.1. What is the speedup of using this new compiler versus using compiler A or B on the original processor?

$T_A/T_{new}=1.67$
$T_{B/}\ T_{new}=2.27$

Question 1.13

Another pitfall cited in Section 1.10 is expecting to improve the overall performance of a computer by improving only one aspect of the computer. Consider a computer running a program that requires 250 s, with 70 s spent executing FP instructions, 85 s executed L/S instructions, and 40 s spent executing branch instructions.

1.13.1 [5] <§1.10> By how much is the total time reduced if the time for FP operations is reduced by 20%?

New CPU Time of FP operations = 56s
New CPU time for the program = 56+85+40+55 = 236
Total time reduced by 5.6%

1.13.2 [5] <§1.10> By how much is the time for INT operations reduced if the total time is reduced by 20%?

T_{new}=250 * 0.8 *=200 s
T_{fp}+ $T_{L/S}$ + T_{branch}= 165s
T_{int} =35s
Reduction time INT :58.8 %

1.13.3 [5] <§1.10> Can the total time can be reduced by 20% by reducing only the time for branch instructions?

Amdahl's

T_{new} **= 250 * 0.8 =200 s**
T_{fp} + T_{Int} + $T_{L/s}$=210 s
No

Question 2.4

For the MIPS assembly instructions above, what is the corresponding C statement? Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively

```
sll  $t0, $s0, 2     # $t0 = f * 4
add  $t0, $s6, $t0   # $t0 = &A[f]
sll  $t1, $s1, 2     # $t1 = g * 4
add  $t1, $s7, $t1   # $t1 = &B[g]
lw   $s0, 0($t0)     # f = A[f]
addi $t2, $t0, 4
lw   $t0, 0($t2)
add  $t0, $t0, $s0
sw   $t0, 0($t1)
```

Answer
B[g] = A[f+1] + A[f]
A and B are 4 byte arrays.

Question 2.12

Assume that registers $s0 and $s1 hold the values 0x80000000 and 0xD0000000, respectively.

2.12.1 [5] <§2.4> What is the value of $t0 for the following assembly code?

add $t0, $s0, $s1

	1	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0
	1	1	0	1		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0
1	0	1	0	1		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0

0x50000000, overflow

2.12.2 [5] <§2.4> Is the result in $t0 the desired result, or has there been overflow?

There has been an overflow

2.12.3 [5] <§2.4> For the contents of registers $s0 and $s1 as specified above, what is the value of $t0 for the following assembly code?

sub $t0, $s0, $s1

0xB0000000

	1	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0
S1																																	
	1	1	0	1		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0
$s1 two's complement																																	
	0	0	1	1		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0
$s0-$s1																																	
	1	0	1	1		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0

2.12.4 [5] <§2.4> Is the result in $t0 the desired result, or has there been overflow?

There is no overflow. This is because the subtraction of 2 negative numbers is a number that is within these 2 numbers (due to negation).

2.12.5 [5] <§2.4> For the contents of registers $s0 and $s1 as specified above, what is the value of $t0 for the following assembly code

```
add $t0, $s0, $s1
add $t0, $t0, $s0
```

$t0=0xD0000000

2.12.6 Is the result in $t0 the desired result, or has there been overflow

There is overflow with the first statement.

Question 2.18

Assume that we would like to expand the MIPS register file to 128 registers and expand the instruction set to contain four times as many instructions.

2.18.1 [5] <§2.5> How this would this affect the size of each of the bit fields in the R-type instructions?
R type instruction is of the format:

Op(6 bits)	rs(5bits)	rt (5bits)	rd (5bits)	shamt(5 bits)	Funct (6 bits)

- The registers need to hold 7 bits of data to support 128 registers. Therefore, rs, rt, rd should be increased from 5 bits to 7 bits.
- Thus, shift amount and function will have to have less number of bits to accommodate the increase in size of rs, rt and rd.
- There are around 50 instructions in the current MIPS-32 instruction set. 4 times as much becomes around 200 instructions. Hence the size of the opcode should be increased from 6 bits to 8 bits.
- To accommodate this, shamt and funct fields will have to reduce its size by 4 bits each, or 8 bits totally.

The new format will look like:

Op (8 bits)	rs (7 bits)	rt (7 bits)	rd (7 bits)	shamt(1 bit)	Funct (2bit)

Or

Op (8 bits)	rs (7 bits)	rt (7 bits)	rd (7 bits)	shamt(2 bit)	Funct (1bit)

2.18.2 [5] <§2.5> How this would this affect the size of each of the bit fields in the I-type instructions?
I type instruction is of the format

Op (6 bit)	rs (5 bit)	rt(5 bit)	Immediate (16 bit)

- Op will need to be increased to 8 bits to accommodate more instructions in the instruction set.
- rs and rt registers will need to be increased to 7 bits to support 128 registers.
- Immediate field will be reduced from 16 bits to 10 bits

The new format will look like:

Op (8 bit)	rs (7 bit)	rt(7 bit)	Immediate (10 bit)

2.18.3 [5] <§§2.5, 2.10> How could each of two propose changes decrease the size of an MIPS assembly program? On the other hand, how could the proposed change increase the size of an MIPS assembly program?
More registers -> more bits per instruction -> could increase code size
More registers-> less register spliss-> less instruction
More instructions-> more appropriate instruction-> decrease code size
More instructions->larger opcodes->larger code size

Question 2.46

Assume for a given processor the CPI of arithmetic instructions is 1, the CPI of load/store instructions is 10, and the CPI of branch instructions is 3. Assume a program has the following instruction breakdowns: 500 million arithmetic instructions, 300 million load/store instructions, 100 million branch instructions
2.46.1 [5] <§2.19> Suppose that new, more powerful arithmetic instructions are added to the instruction set. On average, through the use of these more powerful arithmetic instructions, we can

reduce the number of arithmetic instructions needed to execute a program by 25%, and the cost of increasing the clock cycle time by only 10%. Is this a good design choice? Why?
Avg CPI time = 0.55*1 + 0.33*10 + 0.11*3 = 4.18
Avg CPI time after reduction = 0.48 *1 + 0.38*10 + 0.12 * 3 = 4.11
CPU Time $_{new}$ = .75 *IC_{old} *CPI_A *1.1 *CCT_{old} + IC_{old} *CPI_B *1.1 *CCT_{old}
The gains are offset by the loss.

The extra clock cycle time adds sufficiently to the new CPU time such that it is not quicker than the old execution time in all cases

2.46.2 [5] <§2.19> Suppose that we find a way to double the performance of arithmetic instructions. What is the overall speedup of our machine? What if we find a way to improve the performance of arithmetic instructions by 10 times?
Overall speedup of the machine is 107.7%
Overall speed up by increasing the arithmetic instructions 10 times is 113.3%.

Question 2.47

Assume that for a given program 70% of the executed instructions are arithmetic, 10% are load/store, and 20% are branch.
2.47.1 [5] <§2.19> Given the instruction mix and the assumption that an arithmetic instruction requires 2 cycles, a load/store instruction takes 6 cycles, and a branch instruction takes 3 cycles, find the average CPI.
Average CPI = 0.7*2 + 0.1 * 6 + 0.2 * 3 = 2.6

2.47.2 [5] <§2.19> For a 25% improvement in performance, how many cycles, on average, may an arithmetic instruction take if load/store and branch instructions are not improved at all?

We need 0.75 * 2.6 = 1.95 average CPI to get a 25% improvement in performance.
(0.7 * x) + 0.6 + 0.6 = 1.95
0.7*x = 0.75
x = **1.07**

2.47.3 [5] <§2.19> For a 50% improvement in performance, how many cycles, on average, may an arithmetic instruction take if load/store and branch instructions are not improved at all
We need 0.5 * 2.6 = 1.3 average CPI to get a 50% improvement in performance.
(0.7 * x) + 0.6 + 0.6 = 1.3
0.7*x = 0.1
x = **0.143**
