**Problem 3 (1.3.1 – 1.3.3)**

1.3.1  
**a)** perf of P1 = 3G/1.5 = 2 × 10^9 instructions per cycle  
perf of P2 = 2.5G/1 = 2.5 × 10^9 instructions per cycle  
perf of P3 = 4G/2.2 = 1.82 × 10^9 instructions per cycle  
   P2 is best.  
**b)** perf of P1 = 2G/1.2 = 1.67 × 10^9 instructions per cycle  
perf of P2 = 3G/0.8 = 3.75 × 10^9 instructions per cycle  
perf of P3 = 4G/2 = 2 × 10^9 instructions per cycle  
   P2 is best.  

1.3.2  
**a)** cycles for P1 = 3G × 10 = 3 × 10^{10} cycles  
instructions for P1 = 3G × 10/1.5 = 2 × 10^{10} instructions  
cycles for P2 = 2.5G × 10 = 2.5 × 10^{10} cycles  
instructions for P2 = 2.5G × 10/1 = 2.5 × 10^{10} instructions  
cycles for P3 = 4G × 10 = 4 × 10^{10} cycles  
instructions for P3 = 4G × 10/2.2 = 1.82 × 10^{10} instructions  
**b)** cycles for P1 = 2G × 10 = 2 × 10^{10} cycles  
instructions for P1 = 2G × 10/1.2 = 1.67 × 10^{10} instructions  
cycles for P2 = 3G × 10 = 3 × 10^{10} cycles  
instructions for P2 = 3G × 10/0.8 = 3.75 × 10^{10} instructions  
cycles for P3 = 4G × 10 = 4 × 10^{10} cycles  
instructions for P3 = 4G × 10/2 = 2 × 10^{10} instructions  

1.3.3  
new time = 10 × 0.7 = 7s  
**a)** CPI_{new} = CPI_{old} × 1.2, then CPI(P1) = 1.8, CPI(P2) = 1.2, CPI(P3) = 2.6  
f = No. Instr × CPI/time, then  
f(P1) = 20 × 10^9 × 1.8 / 7 = 5.14 GHz  
f(P2) = 25 × 10^9 × 1.2 / 7 = 4.28 GHz  
f(P1) = 18.18 × 10^9 × 2.6 / 7 = 6.75 GHz  
**b)** CPI_{new} = CPI_{old} × 1.2, then CPI(P1) = 1.44, CPI(P2) = 0.96, CPI(P3) = 2.4
f = No. Instr × CPI/time, then
f(P1) = \(16.66 \times 10^9 \times 1.44/7 = 3.42\) GHz
f(P2) = \(37.5 \times 10^9 \times 0.96/7 = 5.14\) GHz
f(P3) = \(20 \times 10^9 \times 2.4/7 = 6.85\) GHz

**Problem 4 (1.4.1 – 1.4.3)**

1.4.1

Class A: 105 instr.
Class B: 2 × 105 instr.
Class C: 5 × 105 instr.
Class D: 2 × 105 instr.

Time = No. instr × CPI/clock rate

a) Total time P1 = \((10^5 + 2 \times 10^5 \times 2 + 5 \times 10^5 \times 3)/(2.5 \times 10^9) = 10.4 \times 10^{-4}\) s
b) Total time P2 = \((10^5 \times 2 + 2 \times 10^5 \times 2 + 5 \times 10^5 \times 2)/(3 \times 10^9) = 6.66 \times 10^{-4}\) s

1.4.2

CPI = time × clock rate/No. instr

a) 
CPI (P1) = \(2.6\)
CPI (P2) = \(2.0\)

b) 
CPI (P1) = \(1.7\)
CPI (P2) = \(1.2\)

1.4.3

a) 
Clock cycles (P1) = \(26 \times 10^5\)
Clock cycles (P2) = \(20 \times 10^5\)

b) 
Clock cycles (P1) = \(17 \times 10^5\)
Clock cycles (P2) = \(12 \times 10^5\)

**Problem 5 (1.4.4 – 1.4.6)**

1.4.4

a) \((650 \times 1 + 100 \times 5 + 600 \times 5 + 50 \times 2) / (2 \times 10^9) = 2,125\) ns
b) \((750 \times 1 + 250 \times 5 + 500 \times 5 + 500 \times 2) / (2 \times 10^9) = 2,750\) ns

1.4.5

CPI = time × clock rate/No. instr

a) 
CPI = \(3.03\)

b) 
CPI = \(2.75\)

1.4.6

Speedup = \(2,125\) ns/\(2,750\) ns = 0.77
Speedup = \(2,125\) ns/\(2,750\) ns = 0.77

**Problem 6 (1.6.1 – 1.6.3)**

1.6.1

CPI = \(T_{\text{exec}} \times f/\text{No. Instr}\)

a) 
CPI for compiler A = \(1.8\)
CP for compiler B = \(1.8\)

b) 
CPI for compiler A = \(1.1\)
CP for compiler B = \(1.5\)

1.6.2

\(f_A/f_B = (\text{No. Instr}(A) \times \text{CPI}(A))/(\text{No. Instr}(B) \times \text{CPI}(B))\)

a) 
\(f_A/f_B = (1 \times 10^9 \times 1.8)/(1.2 \times 10^9 \times 1.5) = 1.00\)

b) 
\(f_A/f_B = (1 \times 10^9 \times 1.1)/(1.2 \times 10^9 \times 1.25) = 0.73\)

1.6.3

\(T_{\text{new}}/T_{\text{old}} = (I_{\text{count}}(\text{new}) \times \text{CPI}(\text{new}))/(I_{\text{count}}(\text{old}) \times \text{CPI}(\text{old}))\)

a) 
\(T_{\text{new}}/T_A = (0.6 \times 10^9 \times 1.1)/(1 \times 10^9 \times 1.8) = 0.37\)

\(T_{\text{new}}/T_B = (0.6 \times 10^9 \times 1.1)/(1.2 \times 10^9 \times 1.5) = 0.37\)
a) \[ \frac{T_{\text{new}}}{T_A} = \frac{(0.6 \times 10^9 \times 1.1)}{(1 \times 10^9 \times 1.1)} = 0.6 \]
\[ \frac{T_{\text{new}}}{T_B} = \frac{(0.6 \times 10^9 \times 1.1)}{(1.2 \times 10^9 \times 1.25)} = 0.44 \]

**Problem 7 (Only part A of 1.10.1 – 1.10.3)**

1.10.1 For 1 processor system, instruction per processor = 4096, Total instructions = 4096
   - For 1 processor system, instruction per processor = 2046 Total instructions = 4096
   - For 1 processor system, instruction per processor = 1028 Total instructions = 4096
   - For 1 processor system, instruction per processor = 512 Total instructions = 4096

1.10.2 \[ T = \frac{\left[ I_{\text{count}} \text{(Arith)} \times \text{CPI (Arith)} \right] + \left[ I_{\text{count}} \text{(ldst)} \times \text{CPI (ldst)} \right] + \left[ I_{\text{count}} \text{(branch)} \times \text{CPI (branch)} \right]}{f} \]
   - For 1 processor, \[ T = \frac{(2560 \times 1 + 1280 \times 4 + 256 \times 2)}{2 \times 10^9} = 4.096 \text{ microseconds} \]
   - For 2 processor, \[ T = \frac{(1280 \times 1 + 640 \times 5 + 128 \times 2)}{2 \times 10^9} = 2.368 \text{ microseconds} \]
   - For 4 processor, \[ T = \frac{(640 \times 1 + 320 \times 7 + 64 \times 2)}{2 \times 10^9} = 1.504 \text{ microseconds} \]
   - For 8 processor, \[ T = \frac{(320 \times 1 + 160 \times 12 + 32 \times 2)}{2 \times 10^9} = 1.152 \text{ microseconds} \]

1.10.3 For 1 processor, \[ T = \frac{(2560 \times 2 + 1280 \times 4 + 256 \times 2)}{2 \times 10^9} = 5.376 \text{ microseconds} \]
   - For 2 processor, \[ T = \frac{(1280 \times 2 + 640 \times 5 + 128 \times 2)}{2 \times 10^9} = 3.008 \text{ microseconds} \]
   - For 4 processor, \[ T = \frac{(640 \times 2 + 320 \times 7 + 64 \times 2)}{2 \times 10^9} = 1.824 \text{ microseconds} \]
   - For 8 processor, \[ T = \frac{(320 \times 2 + 160 \times 12 + 32 \times 2)}{2 \times 10^9} = 1.312 \text{ microseconds} \]

**Problem 8 (2.10.1 – 2.10.3)**

2.10.1 a) \text{add $s0, $s0, $s0} \\
   b) \text{sub $t1, $t2, $t3} \\

2.10.2 a) \text{R type} \\
   b) \text{R type} \\

2.10.3 a) \text{0x02108020} \\
   b) \text{0x014B4822}

**Problem 9**

\text{sub $t1, $t2, $t3} \\
\text{add $t4, $t1, $t3} \\
\text{sub $t2, $t1, $4}

**Problem 10 (2.13.1 – 2.13.3)(Use modified instructions for 2.13.1)**

2.13.1 a) \text{0xBABEFEF8} \\
   b) \text{0x11DD11D1} \\

2.13.2 a) \text{0xAAAAAA0} \\
   b) \text{0x00DD00D0} \\

2.13.3 a) \text{0x00005545} \\
   b) \text{0x0000BA01}

**Problem 11(Only part B of 2.14.1 – 2.14.3)**

2.14.1 \text{lui $t1, 0x003f} \\
   \text{ori $t1, $t1, 0xffe0} \\
   \text{and $t1, $t0, $t1} \\
   \text{sll $t1, $t1, 9} \\

2.14.2 \text{andi $t0, $t0, 0x000f} \\
   \text{sll $t0, $t0, 14} \\
   \text{ori $t1, $t1, 0x3fff} \\
   \text{sll $t1, $t1, 18} \\
   \text{ori $t1, $t1, 0x3fff} \\
   \text{or $t1, $t1, $t0} \\

2.14.3 \text{srl $t0, $t0, 28} \\
   \text{andi $t0, $t0, 0x0007} \\
   \text{sll $t0, $t0, 14} \\
   \text{ori $t1, $t1, 0x7fff}
sll $t1, $t1, 17
ori $t1, $t1, 0x3fff
or $t1, $t1, $t0

**Problem 12**

add $t2, $zero, 10        # i=10
loopstart: beq $t2, $zero, loopdone    # Jump to end of loop if i == 0
sll $t3, $t2, 4          # $t3 = 16 * i
add $t3, $a0, $t3        # $t3 = address of a[4 * i]
sll $t4, $t2, 3          # $t4 = 8 * i
add $t4, $a1, $t4        # $t4 = address of b[2 * i]
lw $t4, 0($t4)           # $t4 = b[2 * i]
sw $t4, 0($t3)           # a[4 * i] = $t4
sub $t2, $t2, 1          # i--
  j loopstart            # Jump to beginning of loop
loopdone: ...

**Problem 13 (2.39.1 – 2.39.3)**

2.39.1 \( T = \left( (CPI(\text{arith}) \times I_{\text{count}}(\text{arith})) + (CPI(\text{ldst}) \times I_{\text{count}}(\text{ldst})) + (CPI(\text{branch}) \times I_{\text{count}}(\text{branch})) \right) / f \)

a) \( T = (1 \times 500M + 10 \times 300M + 3 \times 100M) / 5G = 0.76s \)
b) \( T = (4 \times 500M + 40 \times 300M + 3 \times 100M) / 5G = 2.86s \)

2.39.2 a) \( T_{\text{new}} = (1 \times 375M + 10 \times 300M + 3 \times 100M) / (5G/1.1) = 0.81s \)
ie, the extra clock cycle time adds sufficiently to the new CPU time such that it is not quicker than the old execution time.

b) \( T_{\text{new}} = (4 \times 375M + 40 \times 300M + 3 \times 100M) / (5G/1.1) = 3.04s \)
ie, the extra clock cycle time adds sufficiently to the new CPU time such that it is not quicker than the old execution time.

2.39.3 a) \( \text{CPI}_{\text{old}} = (1 \times 500M + 10 \times 300M + 3 \times 100M) / 900M = 4.22 \)

\[ \text{CPI}_{\text{new}} \text{ (for doubling arith perf)} = (0.5 \times 500M + 10 \times 300M + 3 \times 100M) / 900M = 3.94 \]
\[ \text{Speedup} \text{ (for doubling arith perf)} = 4.22/3.94 = 107\% \]

\[ \text{CPI}_{\text{new}} \text{ (for 10x arith perf)} = (0.1 \times 500M + 10 \times 300M + 3 \times 100M) / 900M = 3.72 \]
\[ \text{Speedup} \text{ (for 10x arith perf)} = 4.22/3.72 = 113\% \]

b) \( \text{CPI}_{\text{old}} = (4 \times 500M + 40 \times 300M + 3 \times 100M) / 900M = 15.89 \)

\[ \text{CPI}_{\text{new}} \text{ (for doubling arith perf)} = (2 \times 500M + 40 \times 300M + 3 \times 100M) / 900M = 14.78 \]
\[ \text{Speedup} \text{ (for doubling arith perf)} = 15.89/14.78 = 107\% \]

\[ \text{CPI}_{\text{new}} \text{ (for 10x arith perf)} = (0.4 \times 500M + 40 \times 300M + 3 \times 100M) / 900M = 13.89 \]
\[ \text{Speedup} \text{ (for 10x arith perf)} = 15.89/13.89 = 114\% \]

**Problem 14**

objdump is a program for displaying various information about object files. For instance, it can be used as a disassembler to view executable in assembly form. It is part of the GNU binutils for fine-grained control over executable and other binary data.

400474 push %rbp
400475 mov %rsp,%rbp          //Setting up stack registers
400478 movl $0x0,-0x4(%rbp)   //Initialise sum, -0x4(%rbp)
With -O3:
mov $0x114, %eax
retq

The compiler optimizes the loop and determines the sum without executing the loop. Note that 0x114 or 276 in decimal is the sum of integers from 0 to 23!

The other instructions set up context to start the program and are part of standard libraries to load a program.