1.5:

a)
(all answers are in instructions/s).
P1 has 3x10^9/1.5 = 2x10^9
P2 has 2.5x10^9/1 = 2.5x10^9
P3 has 4x10^9/2.2 = 1.8x10^9
Therefore, P2 has the highest performance in instructions per s.

b)
For P1: 3x10^9x10 = 3x10^10 cycles.
For P2: 2.5x10^9x10 = 2.5x10^10 cycles.
For P3: 4x10^9x10 = 4x10^10 cycles.

For P1: 3x10^10/1.5 = 2x10^10 instructions.
For P2: 2.5x10^10/1 = 2.5x10^10 instructions.
For P3: 4x10^10/2.2 = 1.82x10^10 instructions.

c)
First, new CPIs will be as follows:
P1: CPI = 1.5x1.2 = 1.8
P2: CPI = 1.5x1.2 = 1.2
P3: CPI = 1.5x1.2 = 2.64
Clock rates will then be, in instructions, the following:
P1: 2x10^10x1.8/7 = 5.14 GHz
P1: 2.5x10^10x1.2/7 = 4.28 GHz
P1: 1.82x10^10x2.64/7 = 6.86 GHz

1.7

Instruction counts are as follows:
Class A: 10%xE6 = 10^5
Class B: 20%xE6 = 2x10^5
Class C: 50%xE6 = 5x10^5
Class D: 20%xE6 = 2x10^5

Times are as follows:
P1: (10^5 + 2x2x10^5 + 5x10^5 + 2x10^5)/2.5GHz = 1.04x10^-3 s
P2: (2x10^5 + 2x2x10^5 + 5x2x10^5 + 2x2x10^5)/3GHz = 6.67x10^-4 s
Therefore, P2 will be faster.

a)
CPIs are as follows:
P1: 1.04x10^-3x2.5x10^9/10^6 = 2.6
P2: 6.67x10^-4x3x10^9/10^6 = 2.0

b) clock cycles are as follows:
P1: 10^5 + 2x2x10^5 + 5x10^5 + 2x10^5 = 2.6x10^6
P2: 2x10^5 + 2x2x10^5 + 5x2x10^5 + 2x2x10^5 = 2.0x10^6

1.10:

1)
First find the execution time for 1 processor
Exec time|1 processor: (2.56E9 + 12x1.28E9 + 5x256E6) / 2 x E9 = 9.6s

From this, we calculate the speedups for the multiple processors.
2 processors: 0.7 x 2 = 1.4
4 processors: 0.7 x 4 = 2.8
8 processors: 0.7 x 8 = 5.6
Dividing the execution time for 1 processor by speedups, we get the new execution times.
2 processors: 6.85s
4 processors: 3.43s
8 processors: 1.72s

2)
Same method in the first method,
1 processor: (2x2.56E9 + 12x1.28E9 + 5x256E6) / 2 x E9 = 10.9s
Speedups will be the same.
2 processors: 7.77 s
4 processors: 3.89 s
8 processors: 1.94 s

3) We wish to make the execution time using one processor be equal to 3.43s.
Let the CPI be equal to T.
(2.56E9 + Tx1.28E9 + 5x256E6) / 2 x E9 = 3.43s
T = 2.36 i.e. new CPI of load/store should be 2.36.

1.15)

1)
Cycles = 50 x E6 x 1 + 110 x E6 x 1 + 80 x E6 x 4 + 16 x E6 x 2 = 512 x E6
First execution time then is: 512 x E6 / 2 x E9 = 256 x E-3 s.
We wish to improve the CPI of FP instructions to make the program run in 128 x E-3 s.
Let T be the new CPI value for FP instructions.
(50 x E6 x T + 110 x E6 x 1 + 80 x E6 x 4 + 16 x E6 x 2) / 2 = 128 x E-3 s.
T = (256 - 462) / 50 = -4.2.
Obviously, negative CPI is not a possible thing.

2) Same with 1), we first calculate the cycles and the execution time.
Cycles = 50 x E6 x 1 + 110 x E6 x 1 + 80 x E6 x 4 + 16 x E6 x 2 = 512 x E6
First execution time then is: 512 x E6 / 2 x E9 = 256 x E-3 s.
We wish to improve the CPI of FP instructions to make the program run in 128 x E-3 s.
Let T be the new CPI value for the L/S instructions.
(50 x E6 x 1 + 110 x E6 x 1 + 80 x E6 x T + 16 x E6 x 2) / 2 = 128 x E-3 s.
T = 0.8
CPI improvement is, then, 500% = 4/0.8

---

2.7)

8 BYTE WORDS
i->s3
j->s4
A->s6
B->s7

B[8] = A[i] + A[j];

following MIPS code will be equivalent to the above C code:

```
add $t0, $s3, $0    #first store i in t0
add $t2, $s4, $0    #store j in t2
sll $t0, $t0, 3     #multiply t0 by 8
sll $t2, $t2, 3     #multiply t2 by 8
move $t3, $s6       #t3 = A
move $t4, $s6       #t4 = A
add $t3, $t3, $t0    #t3 = &A[i]
add $t4, $t2, $t4    #t4 = &A[j]
move $t1, $s7       #t1 = B
addi $t1, $t1, 64 #t1 = &B[8]
lw $t5, 0($t3)      #store the number in A[i] to t5
lw $t6, 0($t4)      #store the number in A[j] to t6
add $t5, $t5, $t6   #t5 = A[i] + A[j]
sw $t5, 0($t1)
```

2.8)

```
addi $t0, $s6, 4    # t0 = A[1];
add $t1, $s6, $0    # t1 = &A[0];
sw $t1, 0($t0)      # A[1] = A[0];
lw $t0, 0($t0)      # t0 = &A[0]
add $s0, $t1, $t0   # f = &A[0] + &A[0]
```

Following C code is equivalent to the above MIPS code:
f = 2 * (&A);

2.9)

addi $t0, $s6, 4 00100010110010000000000000000100
opcode: 8, rs: 22, rt: 8, immed: 4
add $t1, $s6, $0 00000010110000000100100000100000
opcode: 0, rs: 22, rt: 0, rd: 9
sw $t1, 0($t0) 10101101000010010000000000000000
opcode: 43, rs: 8, rt: 9, immed: 0
lw $t0, 0($t0) 10001101000010000000000000000000
opcode: 35, rs: 8, rt: 8, immed: 0
add $s0, $t1, $t0 00000000100101000100000000100000
opcode: 0, rs: 0, rt: 8, rd: 16

2.20)

```
lw $t1, 0($s0)
sll $t0, $t1, 4
```

2.25)

```
li $t0, 0           #int i = 0
```

```
FirstL:
   slt $t6, $t0, $s0
   beq $t6, 0, Exit1

   li $t1, 0          #int j = 0
   SecondL:
      slt $t6, $t1, $s1   #if (j >= b) goto Exit2
      beq $t6, 0, Exit2

      add $t2, $t0, $t1   #t2 = i + j
      sll $t3, $t1, 4     #t3 = 16 * t1 (4 * 4 as we assume word is 4 bytes)
      add $t3, $s2, $t3   #t3 = &D[4 * j]
      sw  $t2, 0($t3)    #D[4 * j] = i + j

      add $t1, $t1, 1
      j SecondL
   Exit2:
      add $t0, $t0, 1
      j FirstL

Exit1:
```