Samuel Milner

Assignment 1

1. P1 - 3Ghz - CPI = 1.3
   P2 - 3.5 Ghx - CPI = 1.7
   P3 - 4 Ghz - CPI = 2.1

   a.

   $P1\ IPS = \frac{Instructions}{ClockCycles} \times \frac{ClockCycles}{Seconds} = \frac{ClockRate}{CPI}$

   $P2\ IPS\ \frac{3\times10^9}{1.3} = 2,307,692,308$ Instructions per second

   $P3\ IPS = \frac{3.5\times10^9}{1.7} = 2,058,823,529$ Instructions per second

   $P3\ IPS = \frac{4\times10^9}{2.1} = 1,904,761,905$ Instructions per second

   Has best performance

   b.

   $CPUTime = InstructionCount \times CPI \times ClockCycleTime$

   $CPUTime = \frac{CPUClockCycles}{Clockrate}$

   P1 $10 = IC \times 1.3 \times \frac{1}{3\times10^9} \Rightarrow IC = \frac{10 \times 3\times10^9}{1.3}$

   IC = 2.31×10^{10} Instructions executed

   $10 = \frac{CPUClockCycles}{3\times10^9} \Rightarrow 10 \times 3 \times 10^9 = 3 \times 10^{10}$ cycles to execute

   P2 $IC = \frac{10 \times 3.5\times10^9}{1.7} = 2.05 \times 10^{10}$ Instructions executed

   $10 \times 3.5 \times 10^9 = 3.5 \times 10^{10}$ Cycles to execute

   P3 $IC = \frac{10 \times 4\times10^9}{2.1} = 1.9 \times 10^{10}$ Instructions executed

   $10 \times 4 \times 10^9 = 4 \times 10^{10}$ Cycles to execute

   c.

   $IPS = \frac{ClockRate}{CPI} \Rightarrow 2,307,692,308 \times 2 = 4,615,384,615 \sim 4.6$

   Ghz

2.

Arithmetic - CPI = 1 - 2.56×10^{9} instructions

load/store - CPI = 1.2 - 1.286×10^{9} instructions

Branch - CPI = 5 - 2.56×10^{8} instructions

a.

$$CPUTime = InstructionCount \times CPI \times ClockCycleTime$$

1 processor

$$(2.56 \times 10^{9})(5x10^{-10}) + (1.28 \times 10^{9})(1.2)(5x10^{-10}) + (2.56 \times 10^{8})(5)(5x10^{-10})$$

$$= 1.28 + 0.786 + 6.4$$

= 8.45 sec

2 processor

$$= \frac{1.28}{1.4} + \frac{0.786}{1.4} + 6.4$$

= 7.86 sec

4 processor

$$= \frac{1.28}{2.8} + \frac{0.786}{2.8} + 6.4$$

= 7.13 sec

8 processor

$$= \frac{1.28}{11.2} + \frac{0.786}{11.2} + 6.4$$

= 6.66 sec

b.

$\frac{7.86}{8.45}$ 2 are 1.07x faster than 1

$\frac{7.13}{7.86}$ 4 are 1.10x faster than 2

$\frac{6.58}{7.13}$ 8 are 1.08x faster than 2

c.

$$1.28(2) + 0.786 + 6.4 = 9.75$$

$$\frac{1.28(2)}{1.4} + \frac{0.786}{1.4} + 6.4 = 8.79$$
$$\frac{1.28(2)}{2.8} + \frac{0.786}{2.9} + 6.4 = 7.60$$
$$\frac{1.28(2)}{11.2} + \frac{0.786}{11.2} + 6.4 = 6.70$$

As the number of processors increase, the effects of doubling arithmetic CPI decreased

d.

8.45 - 7.86 = 0.59 second difference from 2 to 1 processors
.786 - 0.59 = 0.196 second decrease needed

$$1.28 \times 10^9 \times CPI \times 5 \times 10^{-10} = 0.196$$
$$0.64\, CPI = 0.196$$
$$new\ CPI = 0.30625$$

3. P1 - 4Ghz - CPI = 0.9, $5x10^9$ instructions
   P2 - 3Ghz - CPI = 0.75, $1x10^9$ instructions

a.

P1 $\frac{4\times10^9}{0.9} = 4,444,444,444\ IPS \Rightarrow \frac{5\times10^9}{IPS} = 1.125\ seconds$

P2 $\frac{3\times10^9}{0.75} = 4,000,000,000\ IPS \Rightarrow \frac{1\times10^9}{IPS} = 0.25\ seconds$

P2 is faster

b.

P1 $\frac{1\times10^9}{IPS} = 0.225\ seconds$

P2 $IC = \frac{.225\times3\times10^9}{.75} = 9x10^8$ instructions

P1 can execute more instructions faster than P2, P2 has higher performance though

c.

P1 $IPS \times 0.4 = 1777777778\ FLOPS$

P2 $IPS \times 0.4 = 1600000000\ FLOPS$

P1 has higher FLOPS, which again doesn't match actual performance

4. 0xABCDEF12

   Assuming one knows binary, you could construct a simple table with binary and hex representations of 0-15. Each hex digit is equal to the 4-digit binary number by it in the table. Then you could easily get the binary representation which is easier to convert to decimal.

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
. . .	. . .

   Doing the above mentioned method we get:
   1010 1011 1100 1101 1110 1111 0001 0010
   Then Convert to decimal:
   $2^{31} + 2^{29} + 2^{27} + 2^{25} + \cdots + 2^{1} = 2882400018$

5.
   addi $t0, $s6, 4 I - Type
   Opcode = 001000, rs = 10110, rt = 01000, imm = 0000000000000100

   add $t1, $s6, $0 R-Type
   Opcode = 000000, rs = 10110, rt = 00000, rd = 01001, shamt = 00000, funct = 100000

   Sw $t1, 8($t0) I-Type
   Opcode = 101011, rs = 01000, rt = 01001, imm = 00000000000001000

   Sub $s0, $t1, $t0 R-Type
   Opcode = 000000, rs = 01001, rt = 01000, rd = 10000, shamt = 00000, funt = 100010

6.
   0000 0010 0001 0000 1000 0000 0000 0010 0000
   Opcode = 000000, rs = 10000, rt = 10000, rd = 10000, shamt = 00000, funct = 100000
   Add $s0, $s0, $s0

7.

a. op=0, rs=3, rt=2, rd=3, shamt=0, funct=0x22
   sub $v1, $v1, $v1, Hex: 0x00621822
b. op=0x23, rs=1, rt=2, const=0x4
   lw $v0, 0x0004, $at, Hex: 0x8C220004