General guidelines: Always state your assumptions and clearly explain your answers. Please upload your solution document (PDF or TXT) to Canvas.

100/100 points possible – Due Wednesday, October 28th by 11:59 PM through Canvas.

Part 1: Floating-point Representations

Exercise 1:

Given \( = -1.5625 \times 10^{-1} = 0.15625_{10} \rightarrow \)

\[
\begin{align*}
0.15625 \times 2 &= 0.3125 \quad (0) \\
0.3125 \times 2 &= 0.625 \quad (0) \\
0.625 \times 2 &= 1.25 \quad (1) \\
0.25 \times 2 &= 0.5 \quad (0) \\
0.5 \times 2 &= 1 \quad (1)
\end{align*}
\]

Hence, \( = -0.15625_{10} = -0.00101_{2} \)

\( = -1.01 \times 2^{-3} <\text{Hidden 1} \Rightarrow 1.XXXX> \)

Sign \( = 1 \)
Exponent \( = 15 - 3 = 12 = 01100_{2} \)
Mantissa \( = 0100,0000,00_{2} \)

IEEE Representation: \( 1 \ 01100 \ 0100,0000,00 \)

Exercise 2:

Given \( = (3.984375 \times 10^{-1} + 3.4375 \times 10^{-1}) + 1.771 \times 10^{3} \)

\( = 7.421875 \times 10^{-1} + 1.771 \times 10^{3} \)

\( = 0.0007421875 \times 10^{3} + 1.771 \times 10^{3} = 1.7717421875 \times 10^{3} \)

For IEEE of: \( 1.771 \times 10^{3} = 1771_{10} = 11011101011_{2} = 1.1011,1010,11_{2} \times 10^{10} \)

Sign \( = 0 \)
Exponent \( = 10 = 11001_{2} \)
Mantissa \( = 1011,1010,11_{2} \)
IEEE \( \rightarrow \) \( = 0 \ 11001 \ 1011,1010,11 \)

\( 0.0007421875 \times 10^{3} = 0.7421875_{10} = 0.1011111_{2} = 0.0000,0000,001011111_{2} \times 10^{10} \)

(cont.)
IEEE: 0 11001 1011,1010,11

Part 2: Assembly-level Programming

Exercise 1:

1.a.

$t1 = 10$

LOOP:  
slt $t2, 0, $t1  
beq $t2, 0, DONE
subi $t1, $t1, 1
addi $s2, $s2, 2
j LOOP

DONE:

So, $s2 = 20$

1.b.

temp = true;
while( temp) { OR int i = 10  OR whatever other ideas appear
    temp = 0 < i;
    i = i – 1;
    B = B + 2;
}

1.c.

If $t1 = N$, Complete LOOP ... -> j LOOP happens N times = N*5 instructions
After $t1=0$, 2 steps to get to DONE = 2 instructions

DONE isn’t an instruction, So Total number of instructions = 5*N + 2
Exercise 2:

```
add $t0, $0, $0    # i=0
LOOP1:
  beq $t0, $s0, DONE1   # if (a==i) goto DONE1
  addi $t1, $0, 0    # j=0
LOOP2:
  beq $t1, $s1, DONE2  # if (j==b) goto DONE2
  add $t2, $t0, $t1  # t2 = i+j
  sll $t3, $t1, 4   # t3 = 2^4 * j = 16*j
  add $t4, $s2, $t3  # t4 = address of D + 16*j
  sw $t2, 0($t4)   # *t4 = D[4*j] = t2 = i+j
  addi $t1, $t1, 1   # j++
  j LOOP2   # goto LOOP2
DONE2:
addi $t0, $t0, 1   #i++
j LOOP1    #goto LOOP1
DONE1:
```

Note: The reason why we checked 'i==a' instead of 'i<a' or 'i>a' was to lower the instruction count for “minimum” number of instructions. But it is best to use slt or sgt to check for ‘<’ or ‘>’ before the branch.

Exercise 3:

```
a = 10, b = 1

Complete LOOP2... → j LOOP2, runs 1 time + 1 beq     = 7+1 = 8 instructions
Complete LOOP1... → j LOOP1, runs 10 times + 1 beq     = 10*(4 + 8) + 1 = 121 instructions
```

Total instructions excluding DONE1 because it’s only a label     = 122 instructions
<1 more that initialization before loop1>

Exercise 4:

```
i = 0;   OR i = 0;   OR ????
do {
  s1 = *MemArray;
  result = result + s1;
  MemArray ++;
  i ++;
} while (i<100);
```

```
do {
  s1 = MemArray[i];
  result += result;
  i++; 
}while(i<100)
```
Part 3: Computer Systems Performance

Exercise 1:

1.a.

\[
\text{instruction/time} = \text{instruction/cycle} \times \text{cycle/time} = 1/\text{CPI} \times \text{clock_rate} = \text{clock_rate}/\text{CPI}
\]

P1: \( \frac{3}{1.5} = 2 \text{ E9 instructions/sec} = 2 \times 10^9 \text{ instructions/sec} \)

P2: \( \frac{2.5}{1} = 2.5 \text{ E9 instructions/sec} = 2.5 \times 10^9 \text{ instructions/sec} \)

P3: \( \frac{4}{2.2} \approx 1.82 \text{ E9 instructions/sec} = 1.82 \times 10^9 \text{ instructions/sec} \)

So, P2 has highest performance aka instruction/sec.

1.b.

\[
\# \text{cycles} = \text{clock_rate} \times \text{time} \\
\# \text{instructions} = \text{Cycle/CPI OR [ANS from 1.a]} \times \text{time}
\]

<table>
<thead>
<tr>
<th></th>
<th># cycles</th>
<th># instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>3.0 \times 10^{10}</td>
<td>2 \times 10^{10}</td>
</tr>
<tr>
<td>P2</td>
<td>2.5 \times 10^{10}</td>
<td>2.5 \times 10^{10}</td>
</tr>
<tr>
<td>P3</td>
<td>4.0 \times 10^{10}</td>
<td>1.82 \times 10^{10}</td>
</tr>
</tbody>
</table>

1.c.

\[
\text{clock_rate} = \text{cycle/instruction} \times \text{instruction/time} = \text{CPI} \times \text{instructions/time}
\]

Since the number of instructions is not affected by the change,

\[
\frac{\text{clock_rate}_{\text{new}}}{\text{clock_rate}_{\text{old}}} = \frac{(\text{CPI}_{\text{new}}/\text{time}_{\text{new}})}{(\text{CPI}_{\text{old}}/\text{time}_{\text{old}})}
\]

We have,

- CPI_{new} = 1.2 \times \text{CPI}_{old} (20\% \text{ increase})
- time_{new} = 0.7 \times \text{time}_{old} (30\% \text{ decrease})

So,

\[
\text{clock_rate}_{\text{new}} = \text{clock_rate}_{\text{old}} \times 1.2/0.7 \approx \text{clock_rate}_{\text{old}} \times 1.71
\]

<table>
<thead>
<tr>
<th></th>
<th>3.0GHz \times 1.71 = 5.13 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>2.5GHz \times 1.71 = 4.275 GHz</td>
</tr>
<tr>
<td>P3</td>
<td>4.0GHz \times 1.71 = 6.84 GHz</td>
</tr>
</tbody>
</table>
Exercise 2:

a. Global CPI: <Weighted average of individual CPIs>
   \[
   \text{P1: } (0.1\times1 + 0.2\times2 + 0.5\times3 + 0.2\times3) = 2.6
   \]
   \[
   \text{P2: } 2
   \]

b. faster => more [instruction/time] = more [clock_rate/CPI] \rightarrow From 1.a.
   \[
   \text{P1} = 2.5\text{GHz/CPI} = 2.5/2.6 = 0.96\text{E9 instructions/sec}
   \]
   \[
   \text{P2} = 3\text{GHz/CPI} = 3/2 = 2.00\text{E9 instructions/sec}
   \]
   P2 is faster.

c. Cycles = instruction * global_CPI
   \[
   \text{P1: } 1.0\text{E6}\times2.6 = 2.6\text{E6}
   \]
   \[
   \text{P2: } 1.0\text{E6}\times2 = 2.0\text{E6}
   \]

Exercise 3:

A: 1.0E9 instructions, time = 1.1s
B: 1.2E9 instructions, time = 1.5s

a. Time/cycle = 1ns \rightarrow clock_rate = 1GHz
   CPI = cycle/instruction = (clock_rate*time)/instructions
   A: \((1\text{GHz} \times 1.1\text{s})/1.0\text{E9} = 1.1\)
   B: \((1\text{GHz} \times 1.5\text{s})/1.2\text{E9} = 1.25\)

b. Faster clock for processor => more cycles/sec => more clock_rate
   clock_rate = (CPI*instructions/time)
   [execution time is same for both say t]
   \[
   \text{clock_rate}_A = (1.1 \times 1)/t = 1.1/t
   \]
   \[
   \text{clock_rate}_B = (1.25 \times 1.2)/t = 1.5/t
   \]
   \[
   \text{clock_rate}_A/\text{clock_rate}_B = 1.1/1.5 \sim 0.73
   \]
   Hence, Processor running A is slower by a factor of 0.73 or 27% in terms of clock rate.

Note: you can also do this problem in terms of clock time where you get:
   \[
   \text{clock_time}_A/\text{clock_time}_B = 1.5/1.1 = 1.36
   \]
   So, Processor running A takes 36% more time than that running B.
c. \( C: 6 \times 10^8 = 0.6 \times 10^9 \) instructions, CPI = 1.1, For processor 1.

Execution time = CPI * instruction/clock_rate = 1.1 * 0.6E9 / 1GHz = 0.66 s

[clock_rate remains the same for the same processor as in 1.a]

So, the new compiler is: \( 1.1/0.66 = 1.67 \) times faster than A
And \( 1.5/0.66 = 2.27 \) times faster than B