1. (10%) A loop in C codes is shown as follows.
   
   ```
   while (store[i] == k)
     i = i + 2;
   ```
   
   Assume that `i` and `k` correspond to registers $s3$ and $s5$ and the base of the array `store` is in $s6$. What is the MIPS assembly code corresponding to this C codes?

2. (5%) Suppose you wish to run a program `T` with $8 \times 10^9$ instructions on a 4GHz machine with CPI of 0.9. What is the expected CPU time?

3. (5%) What decimal number is represented by this single precision float?

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. (10%) The following code fragment processes two arrays and produces an important value in register $v0$. Assume that each array consists of 1000 words indexed 0 through 999, that the base addresses of the arrays are stored in $a0$ and $a1$ respectively, and their sizes (1000) are stored in $a2$ and $a3$, respectively.

   ```
   sll $a2, $a2, 2
   sll $a3, $a3, 2
   add $v0, $zero, $zero
   add $t0, $zero, $zero
   outer: add $t4, $a0, $10
   lw $t4, 0($t4)
   add $t1, $zero, $zero
   inner: add $t3, $a1, $t1
   lw $t3, 0($t3)
   bne $t3, $t4, skip
   addi $v0, $v0, 1
   skip: addi $t1, $t1, 4
   bne $t1, $a3, inner
   addi $t0, $t0, 4
   bne $t0, $a2, outer
   ```

   Assume that the above code is run on a machine with 4GHz clock that requires the number of cycles for each instruction shown in Table 1. In the worst case, how many seconds will it take to execute this code?
5. (10%) Consider two different implementations, I1 and I2, of the same instruction set. There are three classes of instructions (X, Y, and Z) in the instruction set. I1 has a clock rate of 8GHz, and I2 has a clock rate of 4GHz. The average number of cycles for each instruction class on I1 and I2 is given in Table 2.

<table>
<thead>
<tr>
<th>Class</th>
<th>CPI on M1</th>
<th>CPI on M2</th>
<th>C1 Usage</th>
<th>C2 Usage</th>
<th>C3 Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>3</td>
<td>2</td>
<td>30</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>Y</td>
<td>4</td>
<td>4</td>
<td>30</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>Z</td>
<td>6</td>
<td>4</td>
<td>40</td>
<td>50</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 2 also contains a summary of average proportion of instruction classes generated by three different compilers. C1 is a compiler produced by the makers of I1, and C2 is a compiler produced by the makers of I2, and the other compiler is a third-party product. Assume that each compiler uses the same number of instructions for a given program but that the instruction mix is as described in Table 2.

(a) Using C1 on both I1 and I2, how much faster can the makers of I1 claim that I1 is compared to I2?

(b) If you purchase I1, which compiler would you use and why?

6. (10%) Convert the following MIPS assembly code to C program. Assume that the floating-point argument IP is passed in $f3 and the result goes in $f0, and $gp is the global pointer.

```assembly
lwcl $f1, const3($gp)
lwcl $f2, const5$(gp)
div.s $f1, $f1, $f2
lwcl $f2, const56($gp)
add.s $f2, $f3, $f2
mul.s $f0, $f1, $f2
jr $ra
```

7. (10%) Suppose we want to execute the following code segment on the pipelined CPU:

```
add $2, $5, $4
add $4, $2, $5
lw $5, 100($2)
add $3, $2, $5
```

Suppose there is no hardware supports for the forwarding and stalls, but the register file can write a register at the first half of a cycle and read it at the second half of the cycle.

(a) How many NOPs and at what places that you can add to make the code segment execute correctly on our pipeline?

(b) Suppose the pipeline stalls, but no forwards, when there is data hazard. How many cycles will the above code segment execute?

8. (20%) Forwarding logic design. For this problem you are to design a forwarding unit for a 5-stage pipeline processor, as shown Figure 1. The forwarding unit returns the value to be forwarded to the current instruction. There are three places that the values for register RS and register RT can come from: decode stage (register file), memory stage, and write-back stage.

```
DECODE STAGE INFORMATION
RS INDEX (5 bits)
RS REG VALUE (32 bits)
MEMORY STAGE INFORMATION
REGISTER INDEX (5 bits)
VALUE (32 bits)
WRITE_ENABLE (1 bit)
WRITE-BACK STAGE INFORMATION
REGISTER INDEX (5 bits)
VALUE (32 bits)
WRITE_ENABLE (1 bit)
FORWARDING UNIT
VALUE FOR RS
VALUE FOR RT
```

Figure 1. Forwarding unit
The write-back and memory stage information consists of: INDEX - explaining which inflight register index is to be written, VALUE - the value that is to be written, ENABLE - whether or not the instruction in the stage is writing.

The decode stage simply states the register index (for RS and RT) and the corresponding register value from the register file.

Generally three values could exist, one of which the forwarding unit should choose for each of the RS and RT register value requests. The memory stage has value MEM, the write-back stage has value WB, and the register file has value RS-REG or RT-REG.

Question: Using the Table 3 below which contains information about all of the instruction stages, indicate which value should be forwarded to the current instruction: MEM, WB, RS-REG, or RT-REG. Each line represents a Forwarding unit evaluation; there is no connection between evaluation lines in the table. You do not need to worry about hazard detection, only value bypassing.

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Mem Stage</th>
<th>Write-Back Stage</th>
<th>Register Stage</th>
<th>RS Value</th>
<th>RT Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Index</td>
<td>Write</td>
<td>Index</td>
<td>Write</td>
<td>RS-Index</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>1</td>
<td>23</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>0</td>
<td>16</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>0</td>
<td>12</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>0</td>
<td>19</td>
<td>0</td>
<td>19</td>
</tr>
</tbody>
</table>

9. (20%) Two important parameters control the performance of a processor: cycle time and cycles per instruction. There is an enduring trade-off between these two parameters in the design process of microprocessors. While some designers prefer to increase the processor frequency at the expense of large CPI, other designers follow a different school of thought in which reducing the CPI comes at the expense of lower processor frequency.

Let us consider the following machines, and compare their performance. Assume the following instruction mix: 25% loads, 13% stores, 47% ALU instruction, 15% branches/jumps.
M1: The multicycle datapath with a 1 GHz clock.
M2: A machine like the multicycle datapath, except that register updates are done in the same clock cycle as a memory read or ALU operation. Thus finite state machine control for multicycle datapath in Figure 2, states 6 and 7 and states 3 and 4 are combined. This machine has a 3.2 GHz clock, since the register update increases the length of the critical path.

M3: A machine like M2 except that effective address calculations are done in the same clock as a memory access. Thus states 2, 3, and 4 can be combined, as can 2 and 5, as well as 6 and 7. This machine has a 2.8 GHz clock because of the long cycle created by combining address calculation and memory access.

(a) Find out which of the machines is fastest in terms of the metric, MIPS.
(b) Are there instruction mixes that would make another machine faster, and if so, what are they?

Figure 2. Finite state machine control for the multicycle datapath.