1. (4 points) MIPS. What is in the specified register at the end of each sequence of code? Give your answer in decimal integers.

```
data
x:       .word    0x8f6ff
y:       .word    0x8000002f
```

(a) Register $t0

```
li $s0,2
lb $t0,x($s0)
```

**ANSWER:** -10

(b) Register $t1

```
lw $s2,y
srl $t1,$s2,1
rol $t1,$t1,2
```

**ANSWER:** 93

2. (6 points) MIPS.

(a) What is the machine language instruction (in hex) for

```
lw    $6,13($12)  ANSWER:  8d86000d
```

(b) What is the assembly language instruction for the following machine language instruction, given in hex:

```
0183502a    ANSWER:  slt  $10, $12, $3
```

3. (5 points) MIPS. Explain how the jump instruction, j, works. Exactly how are the bits of the target address computed? In our multicycle machine design, the jump instruction takes 3 cycles, what happens on each of those cycles? (Only discuss actions relevant to jumps.)

**ANSWER:** The 26 bit target address is taken from the instruction and multiplied by 4 (ie, shifted left 2) to get a word-aligned address. The remaining 4 bits are taken from the high-order bits of the PC. On the first cycle, the instruction is fetched and the PC incremented by 4; on the second cycle the instruction is decoded (ie, it is determined that this is a jump); and in the third step, the address is computed and routed to the PC.

4. (5 points) MIPS. The Fortran compiler does not introduce code to maintain a runtime stack. What does that imply about Fortran routines?

**ANSWER:** They are not recursive (if a routine were to call itself, even indirectly, it would overwrite the saved return address).
5. (5 points) Number Representation. What is the bit pattern that corresponds to 53.375 in IEEE Standard floating point (32-bit, single precision) representation?

**ANSWER:** 0 10000100 101010110000000000000000

6. (6 points) ALU. Regularity is an important design goal. Our ALU design has identical 1-bit ALUs for each of the low-order 31 bits but the high-order ALU is different. In what ways is it different?

**ANSWER:** It has the logic for determining whether or not an overflow had occurred and it has the logic for determining the low-order bit for the slt operation (computed based on the sign bit and the result of overflow detection).

7. (6 points) Single cycle machine. Explain the purpose of the RegDst control line in the single cycle design. Why is it needed? For which instructions (in our implementation set) is it a “don’t care”? For those instructions, what prevents an erroneous register from being written?

**ANSWER:** The RegDst controls which field of the instruction determines the register that is being written. It is used in r-type and lw instructions (set to 1 and 0 respectively). Other instructions do not write the register file (they all set the RegWrite control line to 0).

8. (5 points) Why was the IorD control line needed in the multicycle design but not in the single cycle design?

**ANSWER:** The IorD line controls which address is presented to memory, either the address of the next instruction or the data memory address. In the single cycle design, the instruction and data memories were separate.

9. (7 points) Given the following state machine, construct the truth tables for the NEXT_STATE and OUTPUT functions as defined in class and draw the circuit using the SOP approach as given in class. Use the state labeling as shown. Describe the set of strings for which L is asserted.

![State Machine Diagram]

State: 0
Deassert L

State: 1
Deassert L

State: 2
Assert L

Start

0,1

0

1

0
The output $L$ is asserted on all strings that contain the substring 10.

9. (4 points) Multicycle machine. Our multicycle design had a IRWrite control line that was not present in the single cycle design. Explain why it was needed. Which instructions set it to 1?

ANSWER: In the single cycle design, an instruction was read on each cycle and so there wasn’t any need to store the current instruction. In the multicycle design an instruction was read on the first cycle and its fields were accessed in following cycles so it had to be stored. The IRWrite line insures that the stored instruction is overwritten only on the first cycle of any instruction.

10. (10 points) Performance. Compare our multicycle design $M_m$ and our pipelined design $M_p$ for a program $P$ with the given instruction mix. Assume that 5% of the lw instructions are immediately followed by a use of that value and that 2% of branches are taken. What is the average CPI for each machine? Which machine is faster? How much faster?

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Frequency in P</th>
</tr>
</thead>
<tbody>
<tr>
<td>lw</td>
<td>15%</td>
</tr>
<tr>
<td>sw</td>
<td>10%</td>
</tr>
<tr>
<td>beq</td>
<td>20%</td>
</tr>
<tr>
<td>j</td>
<td>5%</td>
</tr>
<tr>
<td>R-type</td>
<td>50%</td>
</tr>
</tbody>
</table>

CPI $M_m$ = $(.15\times5 + .10\times4 + .20\times3 + .05\times3 + .50\times4) = 3.90$

CPI $M_p$ = $(.15 (.05\times2 + .95\times1) + .10 \times 1 + .20 (.02\times2 + .98\times1) + .05 \times 2 + .50\times1 = 1.06$

These machines have the same architecture and same cycle time, only the CPIs differ. $M_p$ is faster, $3.9/1.06 = 3.68$ times faster
11. (5 points) Performance. For program P and machine Mp in the question above, if the clock rate is 500 MHz, at what MIPS is Mp running P?

ANSWER:

\[(500 \times 10^6 \text{ cycles/sec}) \times (1/1.06 \text{ instructions/cycle}) / 10^6 = 471 \text{ MIPS}\]

12. (8 points) Pipelining. Identify each of the data hazard(s) in the following code sequence. For each, indicate whether or not the hazard could be removed by forwarding.

```
  lw   $s6,39($s2)    ANSWER: There is a hazard between the add
  add  $s2,$10,$s6    and the lw which requires a stall. There is
  sub  $s7,$s6,$s2    a hazard between the sub and the lw which
                    be solved by forwarding, and there is a
                    hazard between the add and sub which will
                    be resolved by forwarding.
```

13. (4 points) Pipelining. Forwarding will solve the hazard(s) in the following code sequence. Identify the hazard(s). In each case, which buffer and field will the value be forwarded from?

```
  lw      $s2,10($s6)
  add     $s2,$s3,$s4
  sub     $s7,$s6,$s2
```

ANSWER: The only hazard is the use of $s2 in the sub which causes problems with both the lw and the add; it will be resolved by forwarding from the EX/MEM buffer (the add) using the destination register field.

14. (3 points) Describe one method of dynamic branch prediction.

ANSWER: Use the low order bits of the instruction address to index into a bit string. Use the accessed bit to predict whether to take the branch, if your guess is wrong, flip the bit.
15. (12 points) Cache. Assume 16-bit addresses, fill in the table and compute the miss rate of the following sequence of references (addresses) for a cache with 128 lines of 8 bytes each.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Hit or Miss?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101 0100 0000 1000</td>
<td>M</td>
</tr>
<tr>
<td>0101 0100 0001 0100</td>
<td>M</td>
</tr>
<tr>
<td>0101 0100 0000 1010</td>
<td>H</td>
</tr>
<tr>
<td>0111 0100 0000 1010</td>
<td>M</td>
</tr>
<tr>
<td>0101 0100 0001 0101</td>
<td>H</td>
</tr>
<tr>
<td>0110 0100 0001 0101</td>
<td>M</td>
</tr>
</tbody>
</table>

16. (5 points) Assume that a machine M has a CPI of 2.3. If 43% of all instructions in program P are loads or stores, how does M with a perfect cache (i.e., no misses) compare to M with a cache having a data miss rate of 6%, an instruction miss rate of 2%, and a miss penalty of 25 cycles. Which is faster? By how much?

ANSWER:

CPI with perfect cache: 2.3
CPI with imperfect cache: 2.3 + .02*25 + .06*.43*25 = 3.445
So the machine with the perfect cache is faster, 3.445/2.3 = 1.5 times faster.