Exam 1 Practice Problems Solution

Note: *The actual exam might have very different problems/questions and might be of a different length. The following problems are intended for you to practice in addition to studying what is in the slides and practicing with the examples in them.*

Problem 1:
Given the following function:

Out = (ab)' + cd' + e'f'

a. Write the expression representing the pull-up network (PUN)
   Out (inputs') = (a'b')' + c'd + ef
   = (a+b) + c'd + ef
b. Write the expression representing the pull-down network (PDN)
   Out' = [(ab)' + cd' + e'f']'
   = (ab) (cd')' (e'f')'
   = ab (c'+d) (e+f)

Problem 2:
We would like to design a circuit that outputs the remainder of dividing its input by 3 (i.e., %3).

a. Fill out the following truth table representing the circuit where X is the input and Y is the output.

X_2	X_1	X_0	Y_1	Y_0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

b. Write the SoP equation for each output bit in terms of the input bits. Do not minimize.

$Y_1 = X_2'X_1X_0' + X_2X_1'X_0$

$Y_0 = X_2'X_1'X_0 + X_2X_1'X_0' + X_2X_1X_0$

Problem 3:

You are to design a component that takes a 3-bit input and outputs the quotient of the division of the input by 3 (this output is 2 bits).

X_2	X_1	X_0	Y_1	Y_0
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0

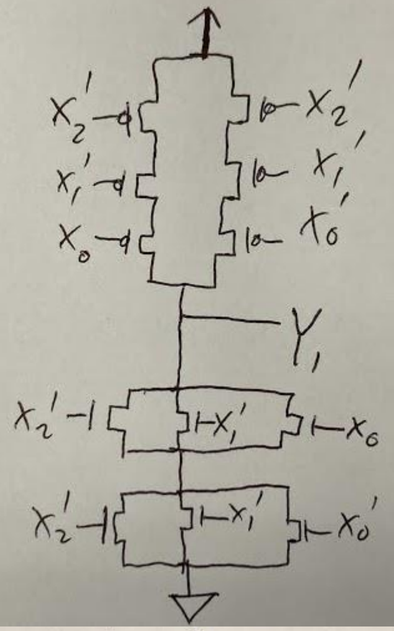
$Y_1 = X_2X_1X_0' + X_2X_1X_0$

$Y_0 = X_2'X_1X_0 + X_2X_1'X_0' + X_2X_1'X_0$

a. Draw the transistor-level circuit for both output bits (do not minimize)

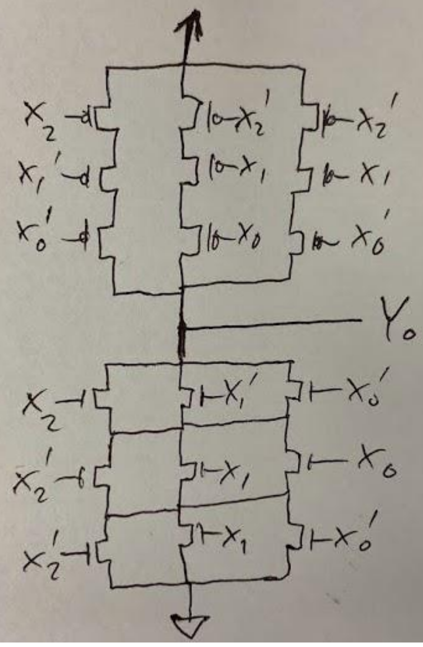
Y_1 PUN:

$Y_1(\text{inputs}') = x_2' x_1' x_0 + x_2' x_1' x_0'$

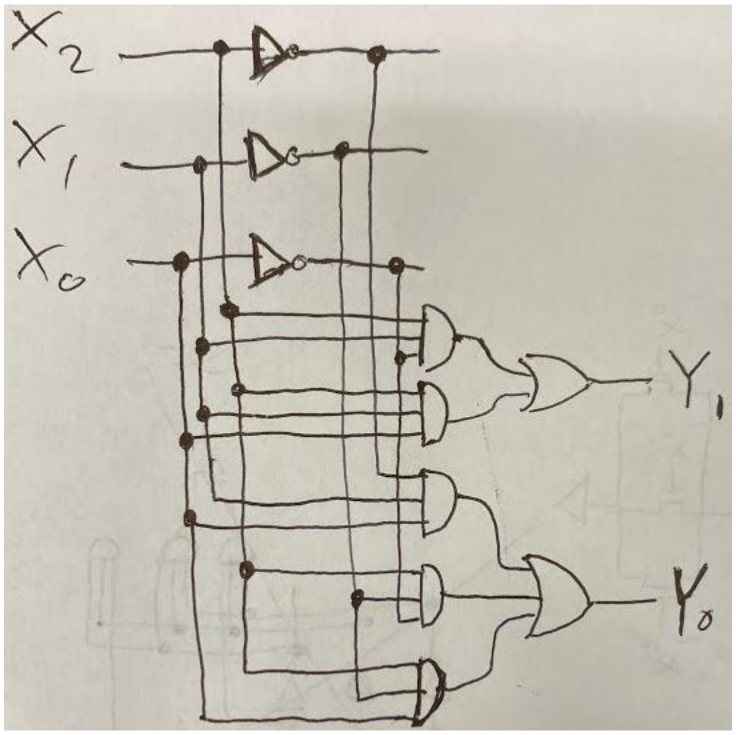


Y_0 PUN:

$Y_0(\text{inputs}') = x_2 x_1' \cancel{x_0}' + x_2' x_1 x_0 + x_2' x_1 x_0'$



b. Draw the gate-level circuit for the component using only NOR gates (start from the SoP and do not minimize)



Then:

- Add 2 bubbles (NOT gates) at the output of every OR gate
- Replace both (OR+NOT) with a NOR
- Add 2 bubbles at the input of every AND gate
- Replace the 5 (NOT+AND) with a NOR
- Replace remaining bubbles/NOT gates with a 2-input NOR gate having its inputs connected

Problem 4:

Write the 6-bit 2's complement representation of the following decimal numbers:

i. +7: 000111

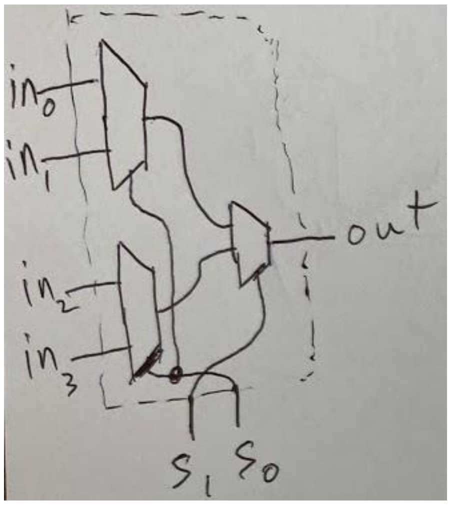
ii. -7: 111001

Problem 5:
Convert the IEEE Single Precision Floating Point 0x42F20000 to binary then to its equivalent decimal value.

1 Sign Bit	8 Biased Exponent Bits	23 Mantissa Bits
0	100 0010 1	111 0010 0000 0000 0000 0000

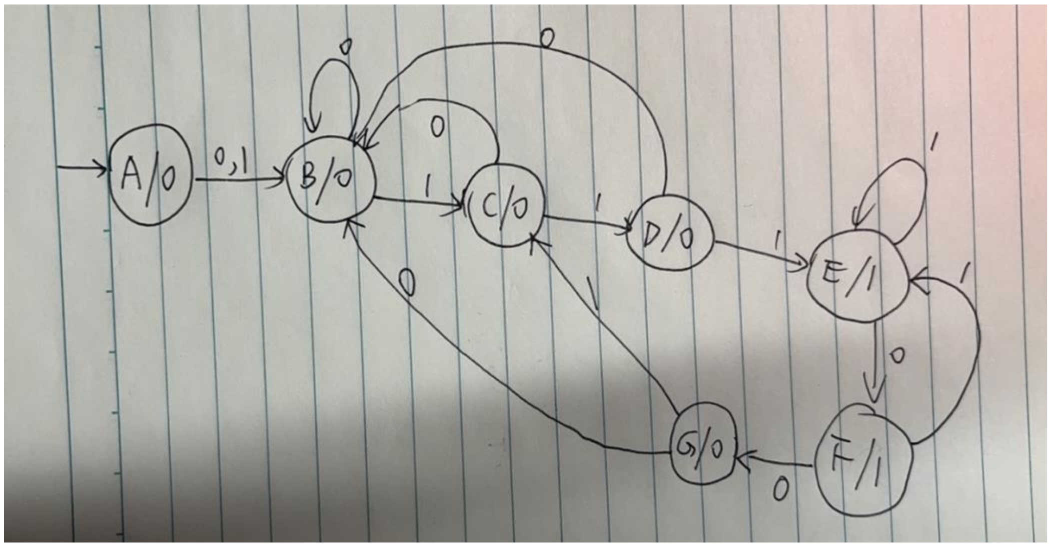
Decimal value:
$+1.111001_2 \times 2^{133-127} = 1111001_2 = 121$

Problem 6:
Draw the design of a 4:1 mux having inputs in0 (top) to in3 (bottom) and select bits s1s0 using 2:1 muxes. Clearly show where each input goes.

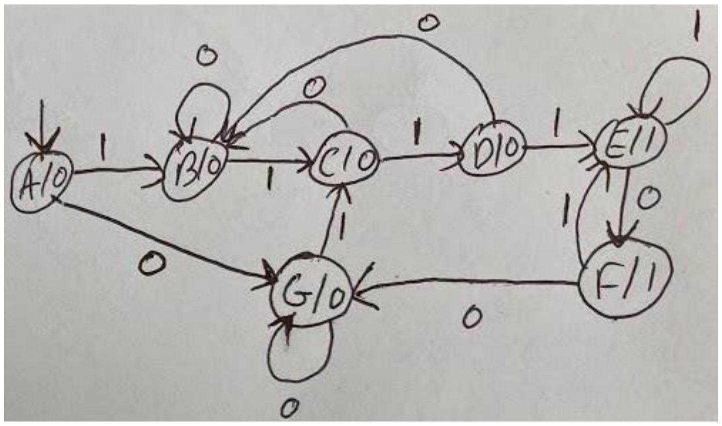


Problem 7:
Draw the Moore state diagram for the FSM with the following specification. The machine has a 1-bit input and a 1-bit output which is initially ‘0’.

- The output of this machine should become ‘1’ if the last 3 out of 4 inputs were ‘1’.
- Once the output switches to ‘1’, it should stay there until 2 consecutive inputs of ‘0’ are received, then it turns ‘0’.



or



Problem 8:
Convert the following C code into MIPS assembly while performing all necessary stack operations. Feel free to use any available registers.

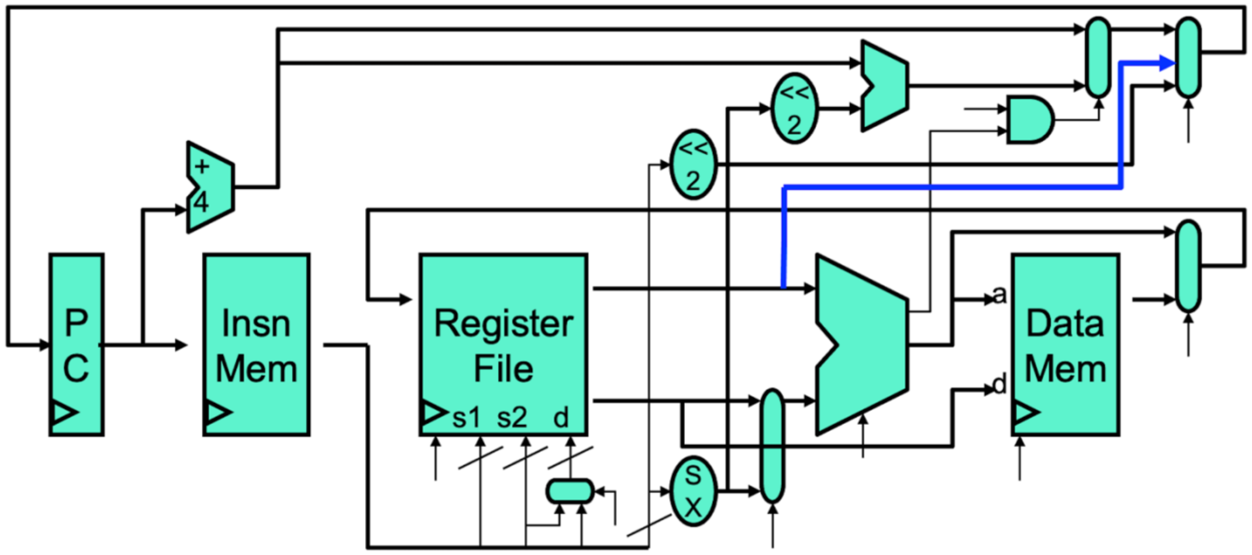
```
int weirdMath(int a, int b) {
      int c = a + b;
      int d = doMagic(c); //function call located at label DO_MAGIC
      c = c + d;
      return c;
}
…
int x = weirdMath(1, 2);
```

```
WEIRD_MATH:      subiu $sp, $sp, 8
                 sw $fp, 0($sp)
                 sw $ra, 4($sp)
                 addiu $fp, $sp, 4
                 add $s0, $a0, $a1
                 add $a0, $s0, $0
                 jal DO_MAGIC
                 add $v0, $s0, $v0
                 lw $ra, 4($sp)
                 lw $fp, 0($sp)
                 addiu $sp, $sp, 8
                 jr $ra
                 …

                 li $a0, 1
                 li $a1, 2
                 jal WEIRD_MATH
```

Problem 9:

Add the necessary components and connections to our 6-instruction MIPS datapath to accommodate for the `jr` instruction.



+
4
P
C
Insn
Mem
Register
File
s1
s2
d
<<
2
<<
2
S
X
a
d
Data
Mem