CS642 Midterm **November 29, 2001** **Answers**

1 a. Shown below is a SRC Assembly language program that is executed. **Next to each instruction that is actually fetched and executed**, identify **all** registers and/or memory locations that are changed by **fetching and executing** the instruction. Show the memory location and register(s} that are changed and and their new contents. Put your answer on the line next to the instruction. If the instruction is not fetched and executed, write NFE. (20 pts.)

```
                                  Location
          .org 1500H
number:   .dc -4                  1500H
          .dw  1                  1504
          .org 2500h
          ldr r0,  number         2500      PC ← 2504,  r0  ← -4
          la  r10, number         2504      PC ← 2508,  r10 ← 1500
          lar r20, routine        2508      PC ← 250C,  r20 ← 2518
          brl r30, r20            250C      PC ← 2518,  r30 ← 2510
          stop                    2510      PC ← 2514,  RUN ← 0
          nop                     2514      NFE
routine:  brpl r30, r0            2518      PC ← 251C
          neg r0, r0              251C      PC ← 2520,  r0 ← 4
          st r0, 4(r10)           2520      PC ← 2524, M(1504) ← 4
          br r30                  2524      PC ← 2510
```

b. In your "blue book", show the hexadecimal for the value of number, the ldr instruction, the lar instruction, and the brpl instruction. See the table of Op-Codes at the bottom of the exam. (20 pts.)

To convert -4 form the 1's complement: 000...0100 → 111...1011, then the 2's complement → 111...1100 = FFFFFFFC
ldr r0, number = 00010 00000 disp. to number (-1004H); -1004 = 1110 1111 1111 1100; sign extend to 22 bits. 0001 0000 00 11 1111 1110 1111 1111 1100 = 103FEFFC
lar r20, routine = 00110 10100 (12 dec. extended to 22 bits) = 0011 0101 0000 0000 0000 0000 0000 1100 = 3500000C
brpl r30, r0 = 01000 00000 11110 00000 (4 in last 3 bits) = 0100 0000 0011 1100 0000 0000 0000 0100 = 403C0004

2 a. Show the truth tables for the Sum (S) and Carry-out (C_{out}) of a Full adder in terms of the two input bits (X, and Y), and the Carry-in (C_{in}). (5 pts.)
b. Draw the Karnaugh Maps for the S and C_{out} and derive minimal sums of products. (10 pts)
c. Show the NAND gate implementations of S and C_{out}. You may assume that X#, Y#, and C_{in}# are also present and need not show inverters to generate them. (10 pts)
d. Implement S using a 4 input MUX. (5 pts.)

X	Y	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S

C_{in} \ XY	00	01	11	10
0	0	1	0	1
1	1	0	1	0

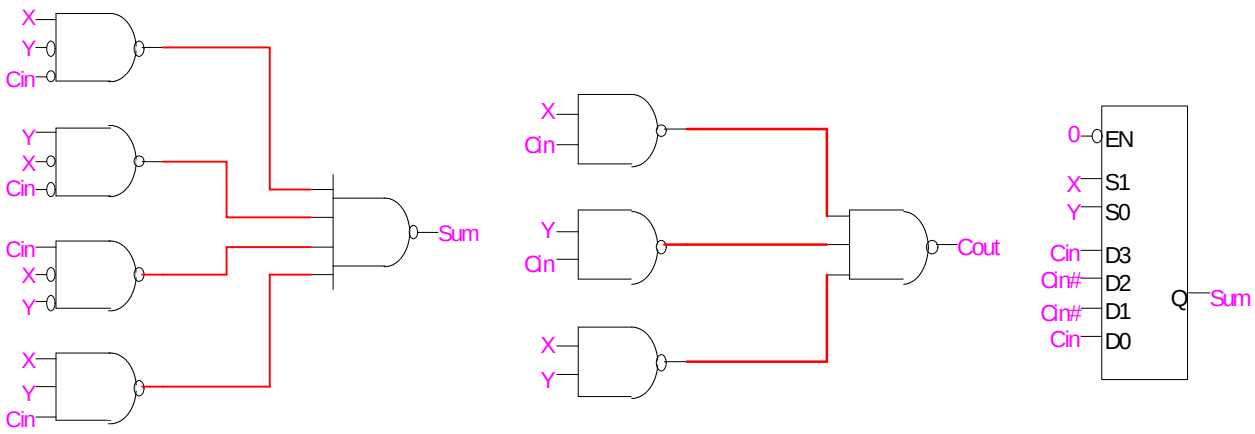
C_{out}

C_{in} \ XY	00	01	11	10
0	0	0	1	0
1	0	1	1	1

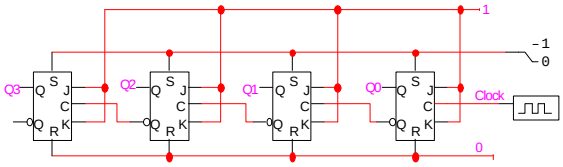
Y Cin X Y X Cin

$S = X\#Y\#C_{in} + X\#YC_{in}\# + XY\#C_{in}\# + XYC_{in}$

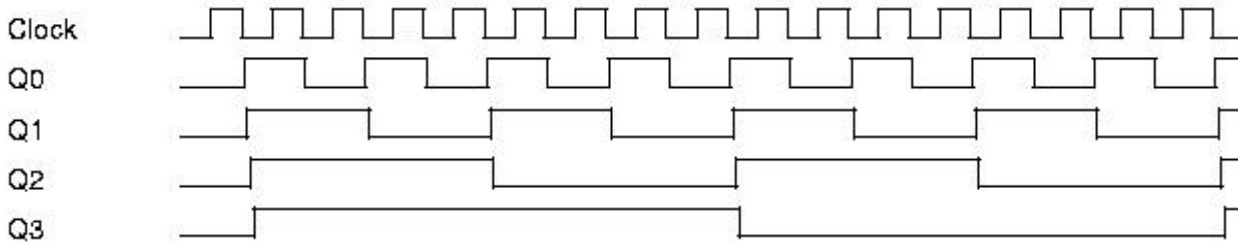
$Cout = XC_{in} + YC_{in} + XY$



3 a. Show how to implement a ripple down counter that will count from 15 - 0. You need not show the gates required to implement the flip-flops (FFs), but you must label all inputs to the FF's. (10 pts.)



b. Draw a timing diagram that shows the the Clock and the outputs of the FF's. Under the bottom most output show the decimal values of the count. Be sure to include all glitches. (10 pts.)



4. The figure below shows a circuit and the input waveforms. **On the diagram**, draw the waveforms of the outputs C and D. You may disregard the gate propagation delays. (10 pts.)

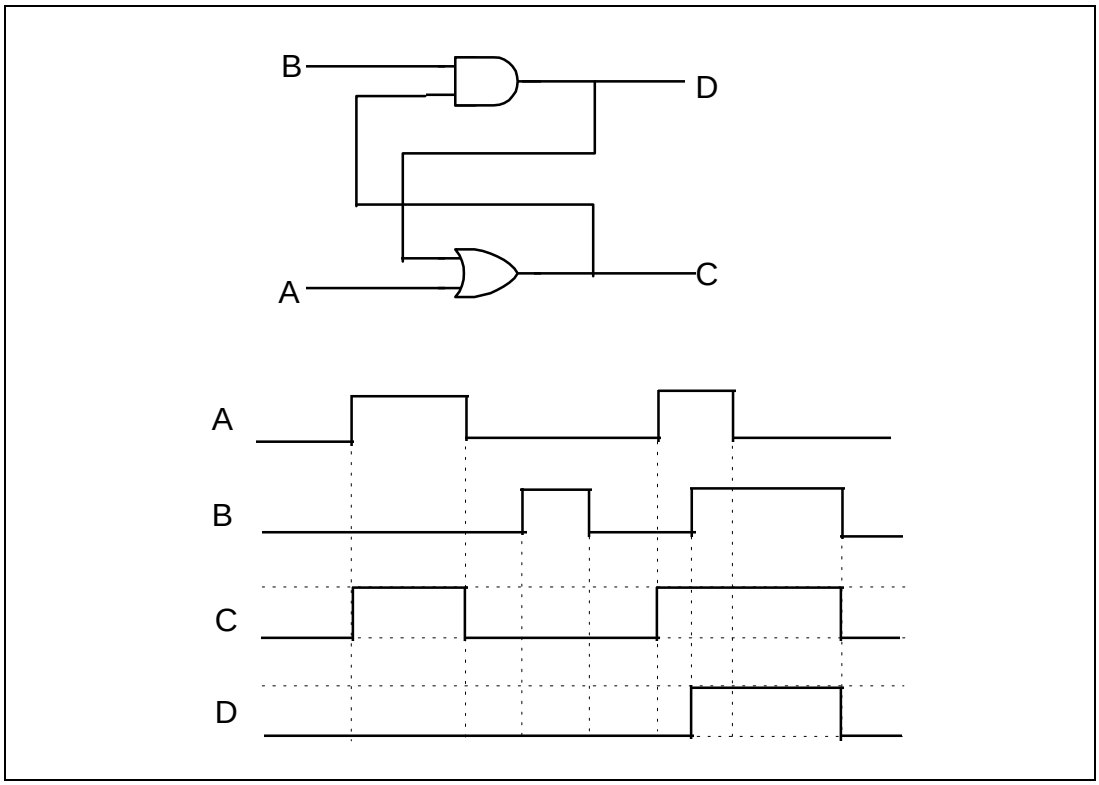


Table of Defined SRC Op-Codes in Decimal

0	**= nop**	**1**	**= ld**	**2**	**= ldr**	**3**	**= st**	**4**	**= str**	**5**	**= la**	**6**	**= lar**	**8**	**= br**
9	**= brl**	**12**	**= add**	**13**	**= addi**	**14**	**= sub**	**15**	**= neg**	**20**	**= and**	**21**	**= andi**	**22**	**= or**
23	**= ori**	**24**	**= not**	**26**	**= shr**	**27**	**= shra**	**28**	**= shl**	**29**	**= shc**	**31**	**= stop**		