Josh Newburn

Part 1 a)

k_1k_0 \ a_i	0	1
00	0	1
01	0	1
11	1	0
10	0	1

a_i^*

k_1k_0 \ b_i	0	1
00	1	1
01	0	1
11	0	0
10	1	0

b_i^*

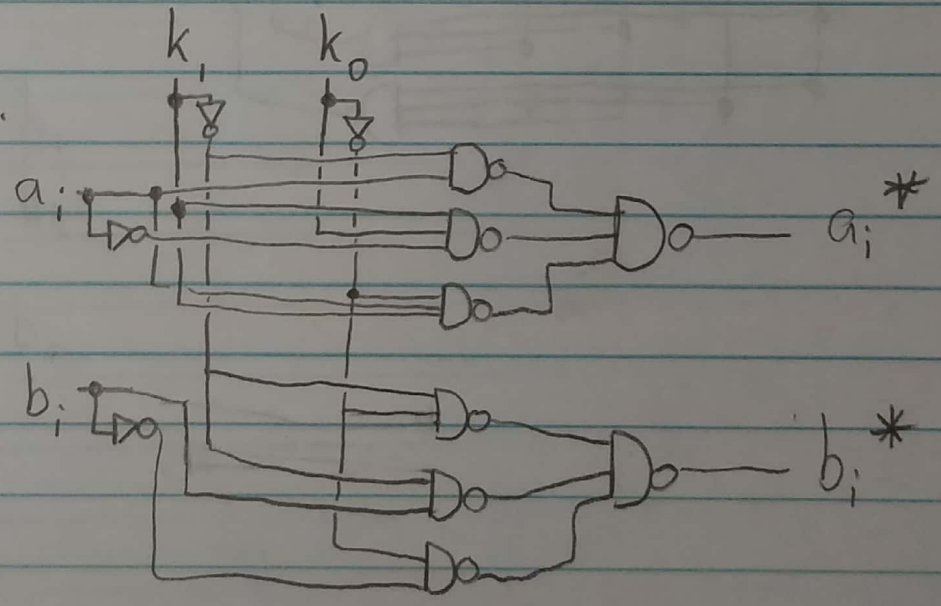
k_1 \ k_0	0	1
0	0	0
1	1	1

c_0

1. $a_i^* = k_1' a_i + k_1 k_0 a_i' + k_1 k_0' a_i$

$b_i^* = k_1' k_0' + k_1' b_i + k_0' b_i'$

2.



3. $c_0 = k_1$

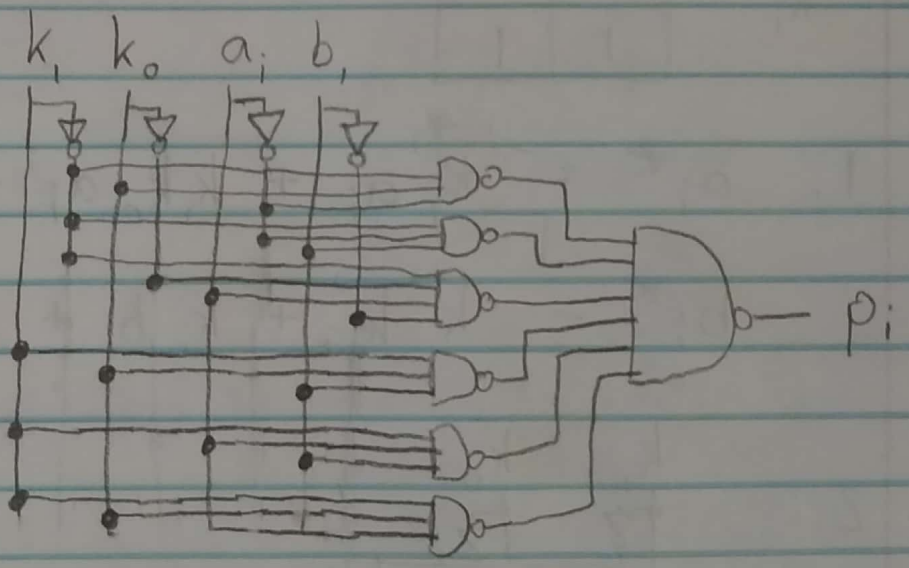
Part 1 b)

p_i

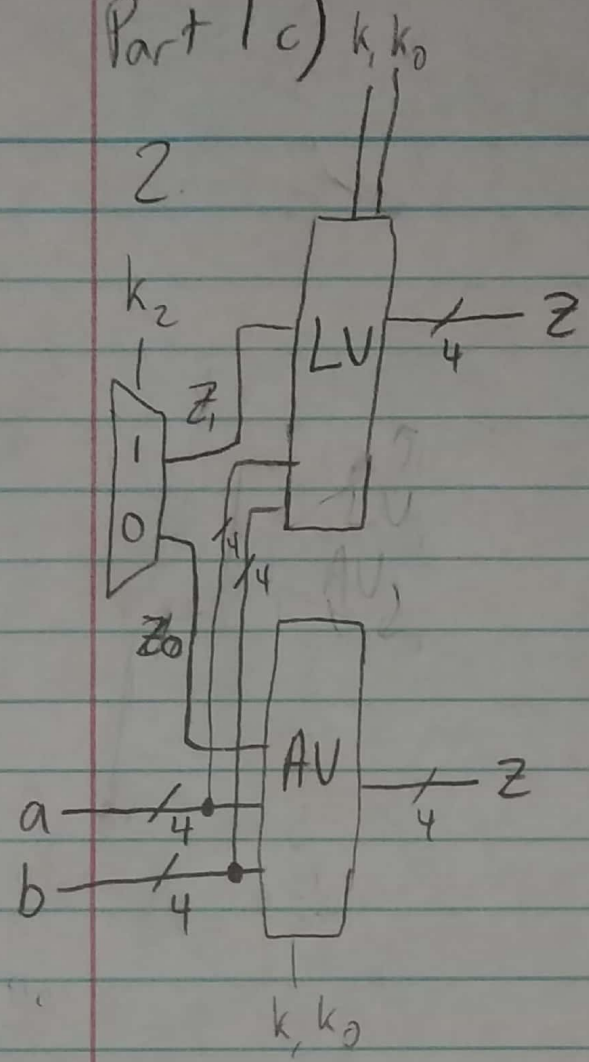
$k_1 k_0$ \ $a_i b_i$	00	01	11	10
00	0	1	0	1
01	1	1	0	0
11	0	1	1	1
10	0	0	1	0

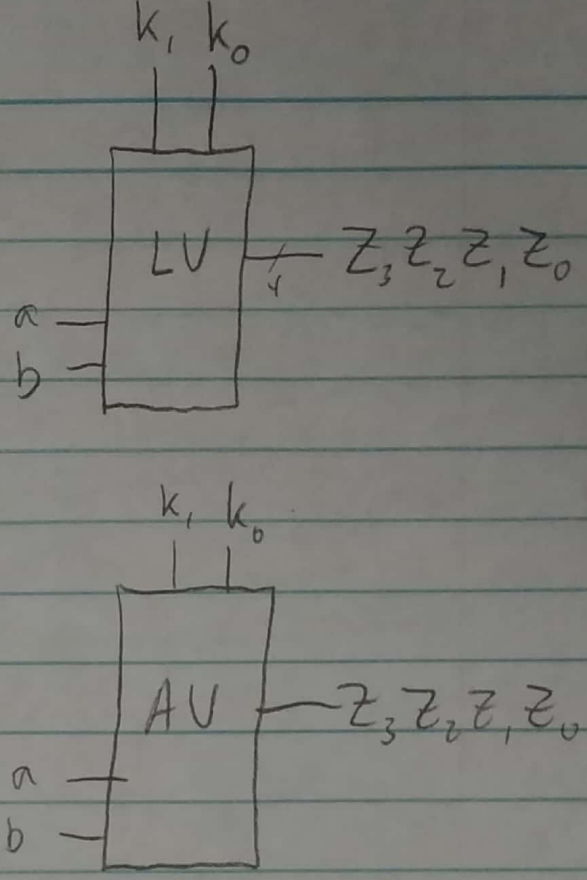
1. $p_i = k_1' k_0 a_i' + a_i' b_i k_1' + k_1' k_0' a_i b_i'$
$+ k_1 k_0 b_i + a_i b_i k_1 + k_1 k_0 a_i$

2.



Part 1 c)





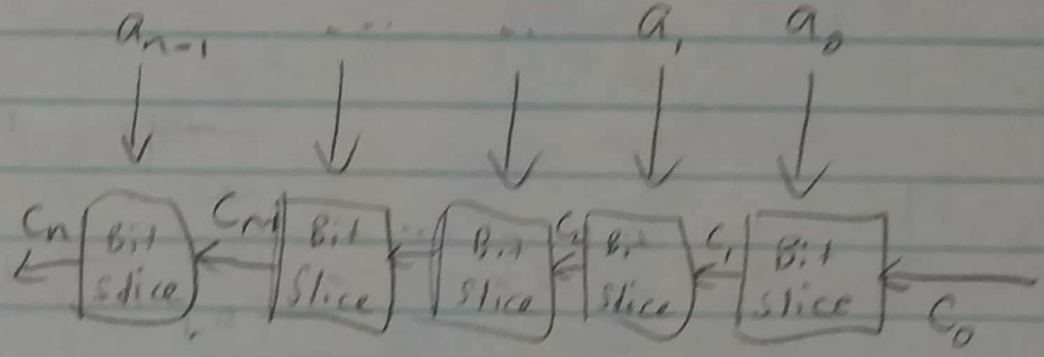
Part 2

c_i	a_i	c_{i+1}	
0	0	0	Past even, now even
0	1	1	Past even, now odd
1	0	1	Past odd, now odd
1	1	0	Past odd, now even

c_i \ a_i	0	1
0	0	1
1	1	0

$c_i a_i' + c_i' a_i = XOR(a_i, c_i)$

1.



2. Inputs: a_i is current digit in binary #
c_i indicates if previous digits have an odd (1) or even (0) number of 1's
c_{i+1} indicates if previous digits including a_i are negative or positive

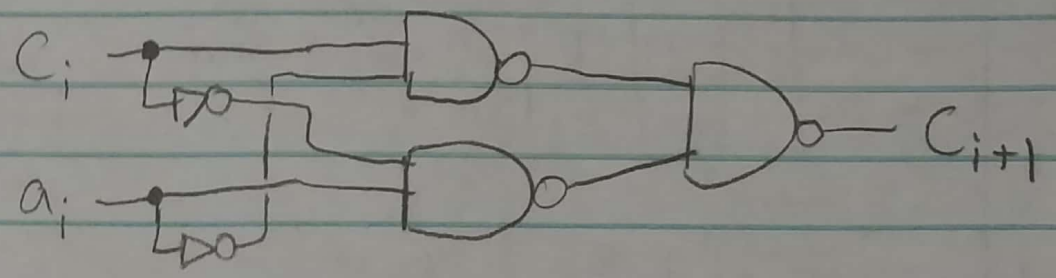
3.

4. $c_{i+1} = c_i a_i' + c_i' a_i$

Part 2 (cont.)

5. $c_i a_i' + c_i' a_i$

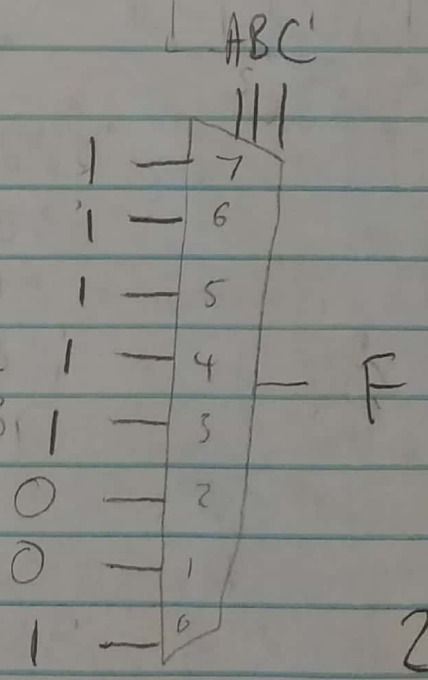
$c_{i+1} = ((c_i a_i')'(c_i' a_i)')'$



Part 3

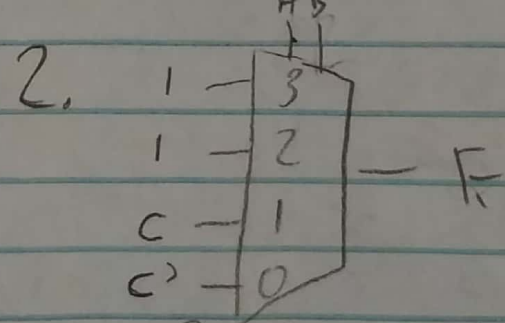
$F(A, B, C)$

1.



ABC	F	
000	1	I_0
001	0	I_1
010	0	I_2
011	1	I_3
100	1	I_4
101	1	I_5
110	1	I_6
111	1	I_7

2.



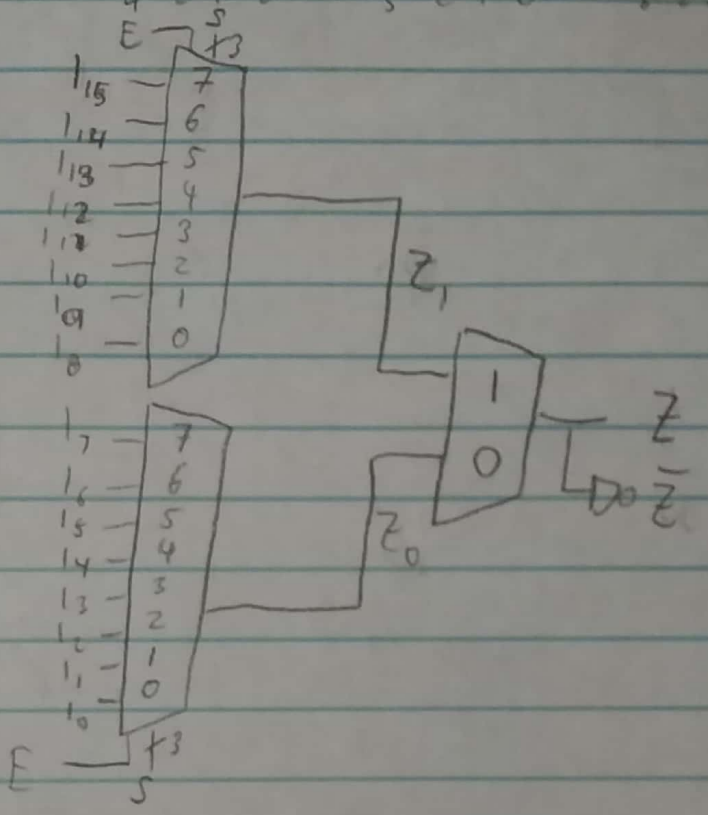
3. Requires 3 bits

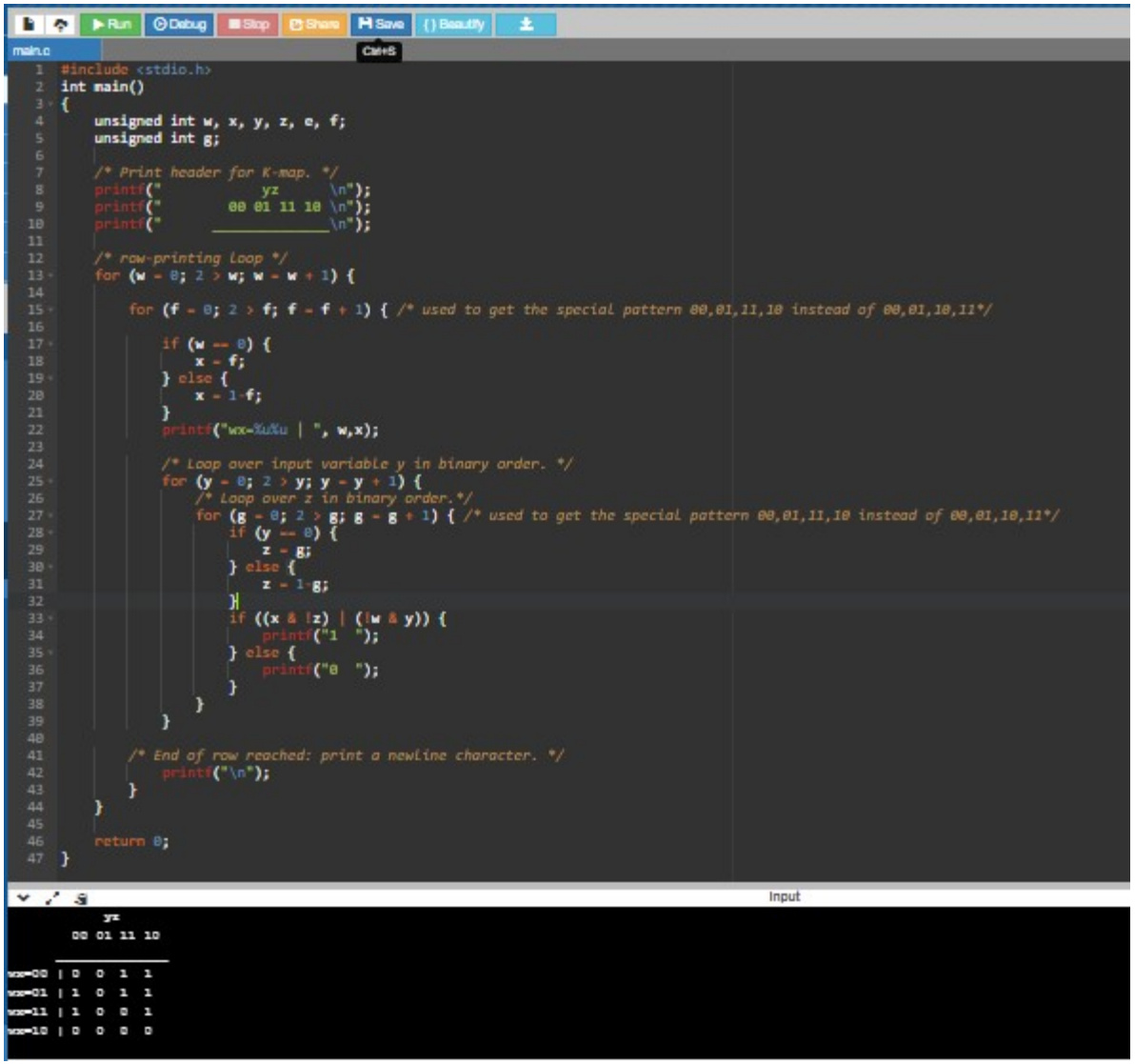
3. Requires 2 bits

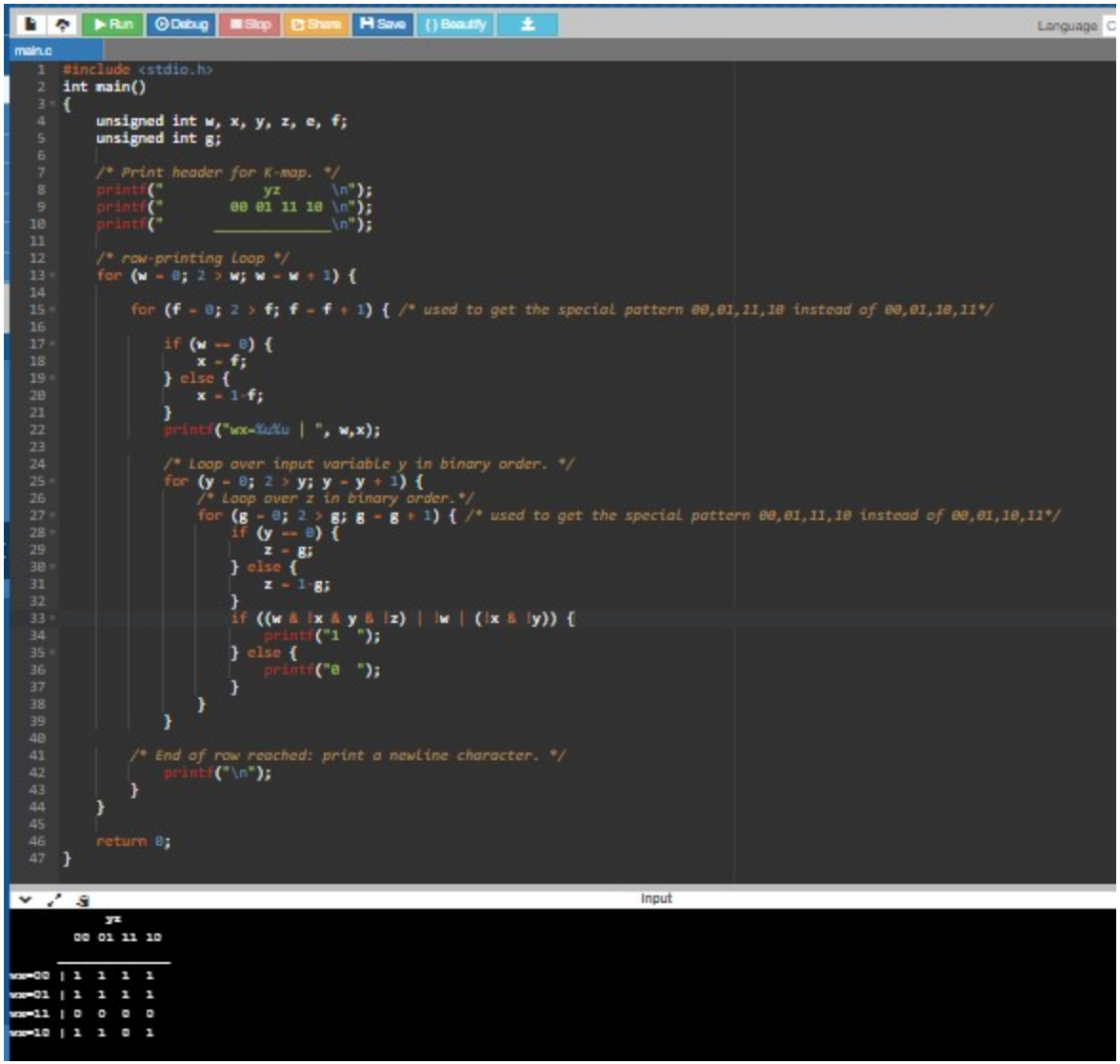
Part 4

1. $Z = I_0 S_2' S_1' S_0' E + I_1 S_2' S_1' S_0 E + I_2 S_2' S_1 S_0' E + I_3 S_2' S_1 S_0 E$
$+ I_4 S_2 S_1' S_0' E + I_5 S_2 S_1' S_0 E + I_6 S_2 S_1 S_0' E + I_7 S_2 S_1 S_0 E$

2.




Run
Debug
Stop
Share
Save
Ctrl+S
Beautify
main.c
#include <stdio.h>
int main()
{
unsigned int w, x, y, z, e, f;
unsigned int g;
/* Print header for K-map. */
printf(" yz \n");
printf(" 00 01 11 10 \n");
printf(" \n");
/* row-printing loop */
for (w = 0; 2 > w; w = w + 1) {
for (f = 0; 2 > f; f = f + 1) { /* used to get the special pattern 00,01,11,10 instead of 00,01,10,11*/
if (w == 0) {
x = f;
} else {
x = 1-f;
}
printf("wx=%u%u | ", w,x);
/* Loop over input variable y in binary order. */
for (y = 0; 2 > y; y = y + 1) {
/* Loop over z in binary order.*/
for (g = 0; 2 > g; g = g + 1) { /* used to get the special pattern 00,01,11,10 instead of 00,01,10,11*/
if (y == 0) {
z = g;
} else {
z = 1-g;
}
if ((x & !z) | (!w & y)) {
printf("1 ");
} else {
printf("0 ");
}
}
}
/* End of row reached: print a newline character. */
printf("\n");
}
}
return 0;
}
Input
yz
00 01 11 10
wx=00 | 0 0 1 1
wx=01 | 1 0 1 1
wx=11 | 1 0 0 1
wx=10 | 0 0 0 0


Run
Debug
Stop
Share
Save
{} Beautify
Language
main.c
#include <stdio.h>
int main()
{
unsigned int w, x, y, z, e, f;
unsigned int g;
/* Print header for K-map. */
printf(" yz \n");
printf(" 00 01 11 10 \n");
printf(" _____\n");
/* row-printing Loop */
for (w = 0; 2 > w; w = w + 1) {
for (f = 0; 2 > f; f = f + 1) { /* used to get the special pattern 00,01,11,10 instead of 00,01,10,11*/
if (w == 0) {
x = f;
} else {
x = 1-f;
}
printf("wx=%u%u | ", w,x);
/* Loop over input variable y in binary order. */
for (y = 0; 2 > y; y = y + 1) {
/* Loop over z in binary order.*/
for (g = 0; 2 > g; g = g + 1) { /* used to get the special pattern 00,01,11,10 instead of 00,01,10,11*/
if (y == 0) {
z = g;
} else {
z = 1-g;
}
if ((w & !x & y & !z) | !w | (!x & !y)) {
printf("1 ");
} else {
printf("0 ");
}
}
}
/* End of row reached: print a newline character. */
printf("\n");
}
}
return 0;
}
Input
yz
00 01 11 10
wx=00 | 1 1 1 1
wx=01 | 1 1 1 1
wx=11 | 0 0 0 0
wx=10 | 1 1 0 1