Buttons and Keypads

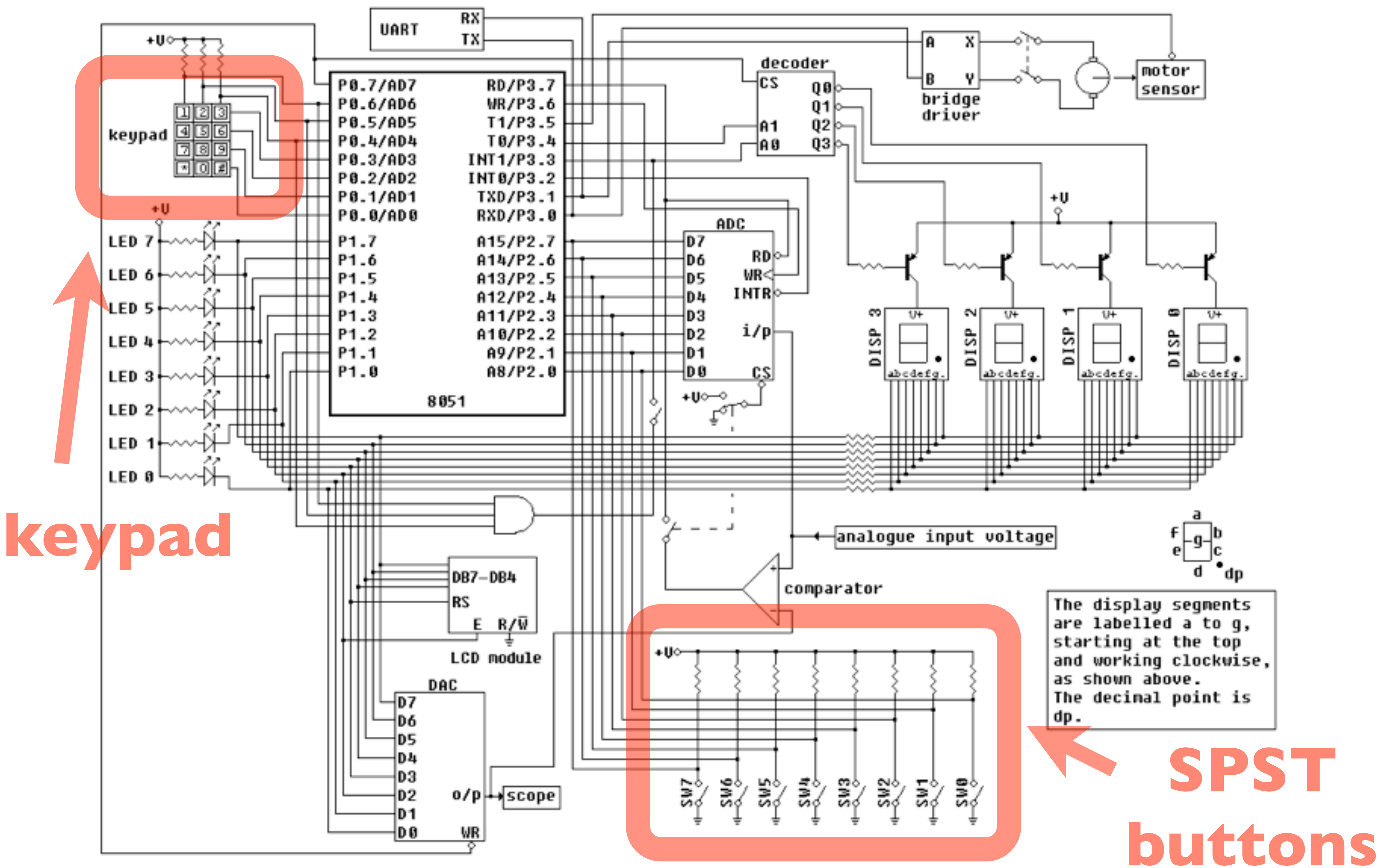
Buttons vs keypad

- Button
  - generates logic 0 and 1 values
  - implemented as a switch w/ pull up/ down
- Keypad
  - conceptually a matrix of buttons
  - Processed by <u>scanning rows & columns</u>

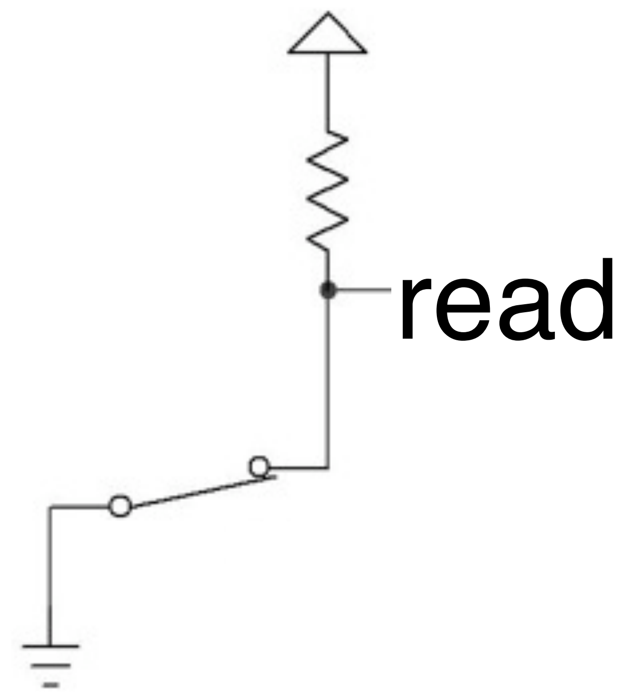
Button

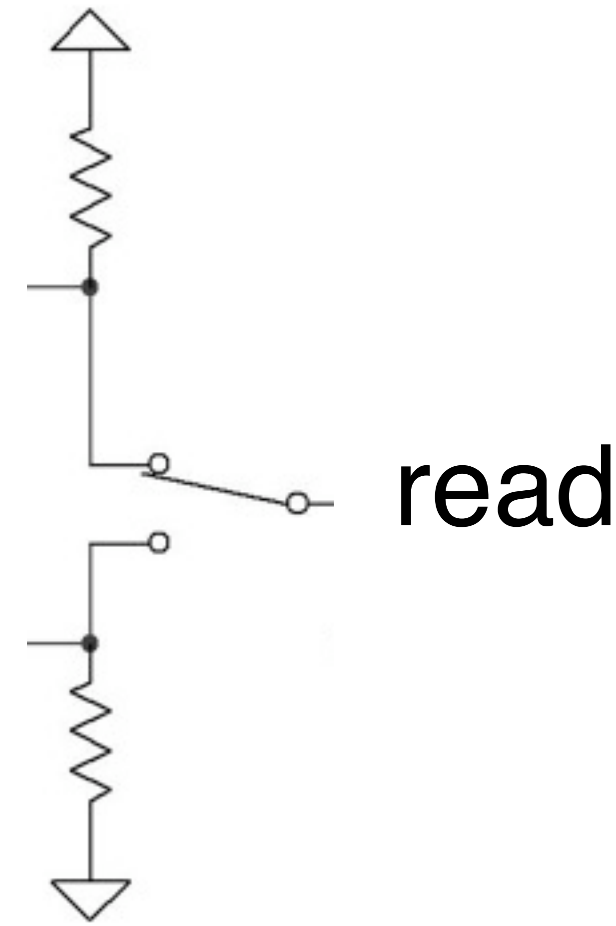
- Two conceptual states: on/off
- Implementation:
  - SPDT switch: connects to logic 0 or 1
  - SPST switch: connects to logic 0 or pull
- Issues
  - power consumption, key bounce

Revisited: EdSim51



SPST vs. SPDT





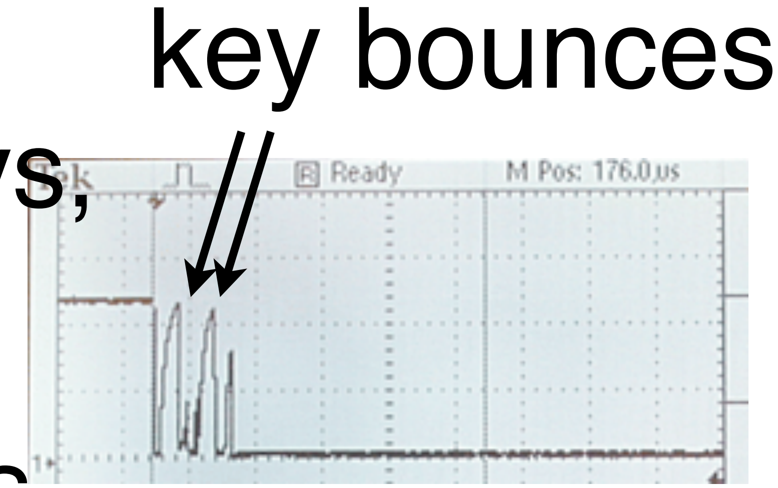
2 states:
pull up or Gnd

3 states:
pull up, pull down, and
in between!

SPST vs. SPDT

- SPST: simpler
  - consumes virtually no current when open
  - consumes some current when pressed
- SPDT
  - consumes same current when open,
  - consumes less current when pressed

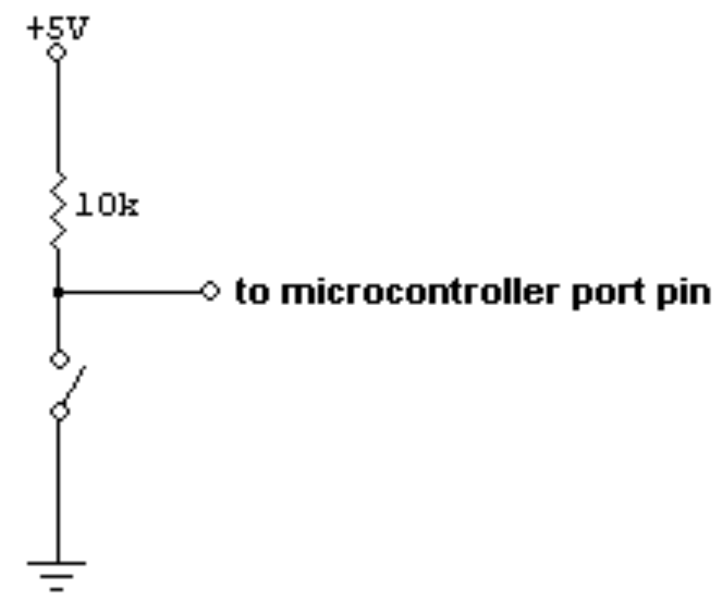
Key bouncing

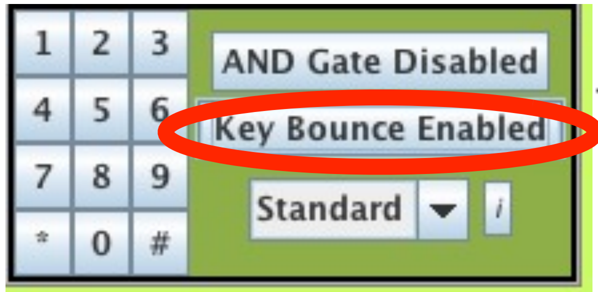


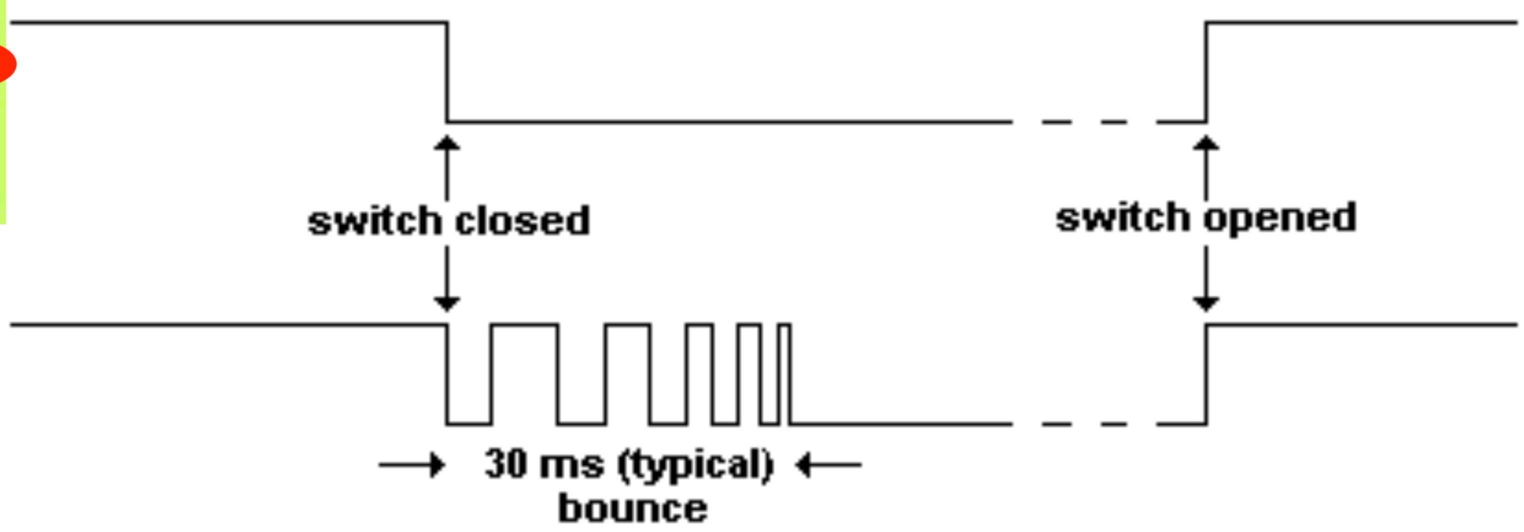
- When you press/release keys, could get key bouncing
    - User may intend one press, but looks like multiple fast presses
- Solutions:
    - hardware: debouncing switch (cross-coupled NANDs as SR latch)
    - software: delay (~20ms)

Key bouncing in EdSim

- Not pressed: pulled high
- Pressed: pulled low
- Can simulate bouncing



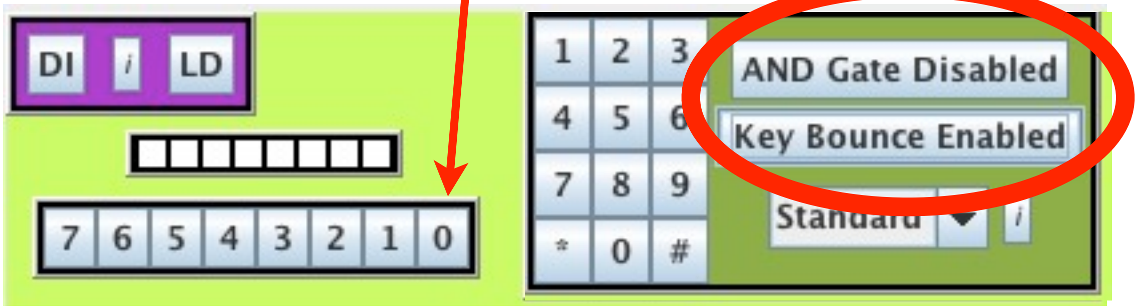




Key bouncing in EdSim

- Test: use switch#0 (P2.0)
- See how many times it increments on each press/release
- Display using LEDs (P1)

P2.0

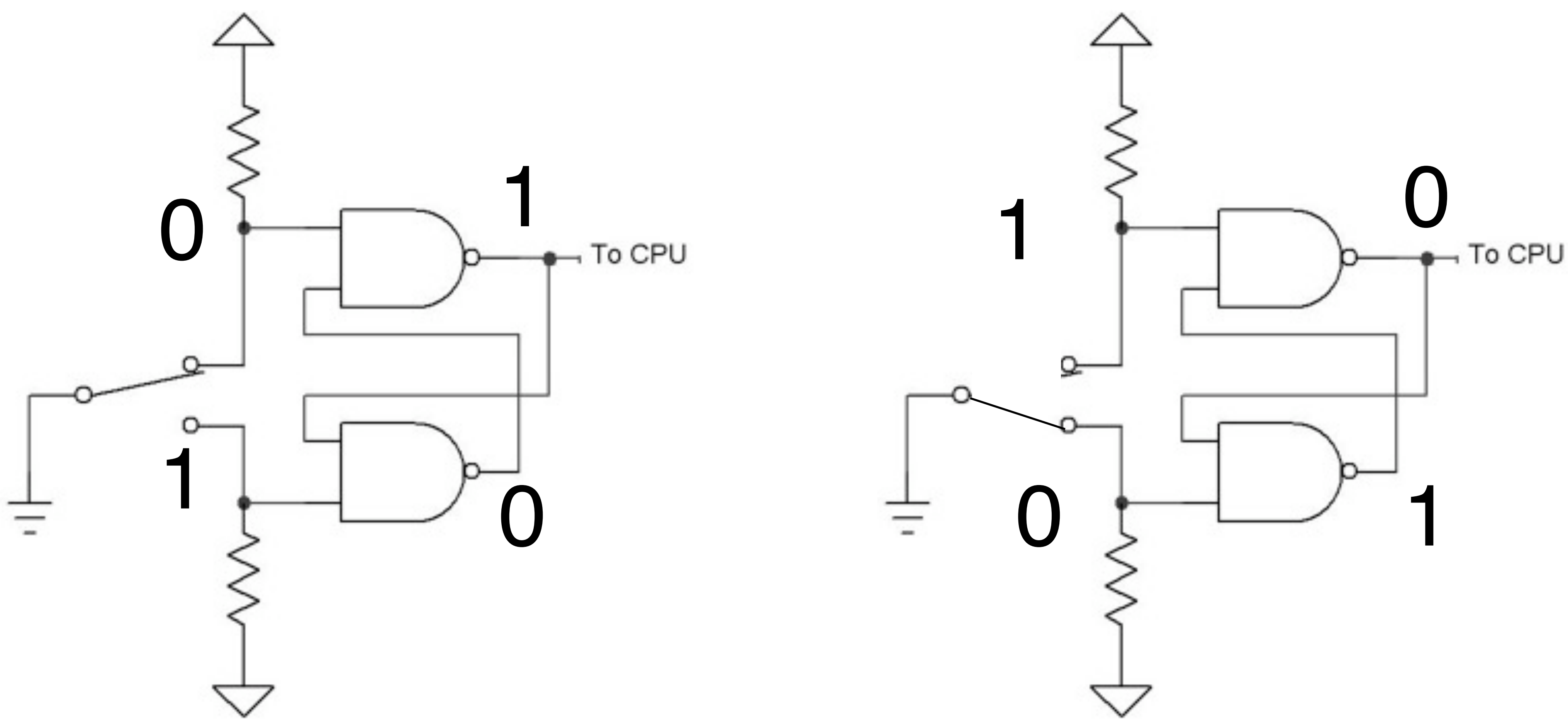


P1

```
notPressed:  JB  P2.0, notPressed
             ;; increment LED bank, which is P1
             INC P1
             ;; now pressed, wait until button release
pressed:
             JNB P2.0, pressed
             ;; now jump back to beginning
             SJMP  notPressed
```

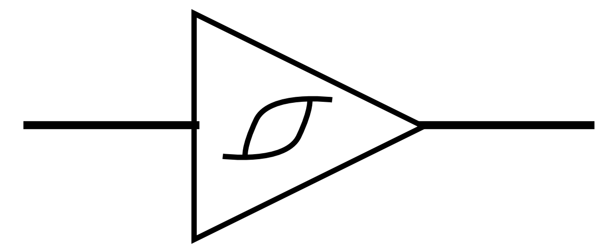
Hardware Debouncer

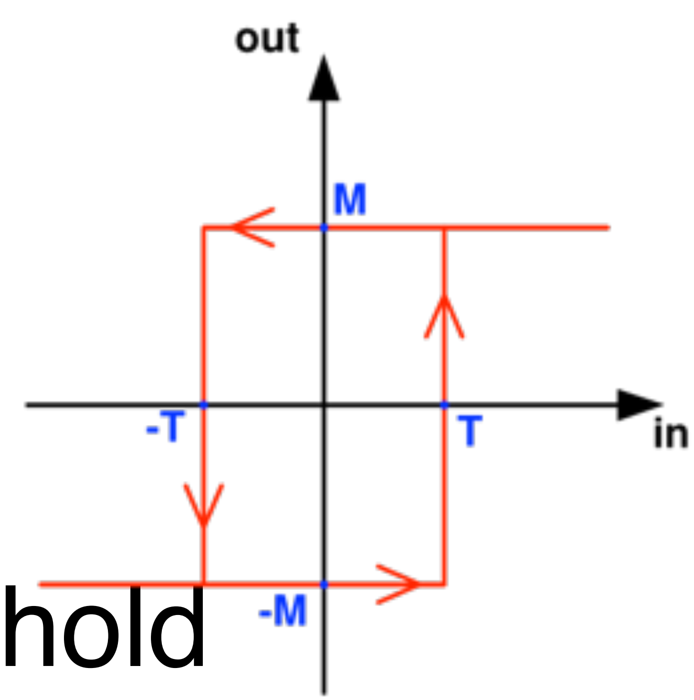
- SR Latch with two steady states



Debouncing with Schmitt Trigger

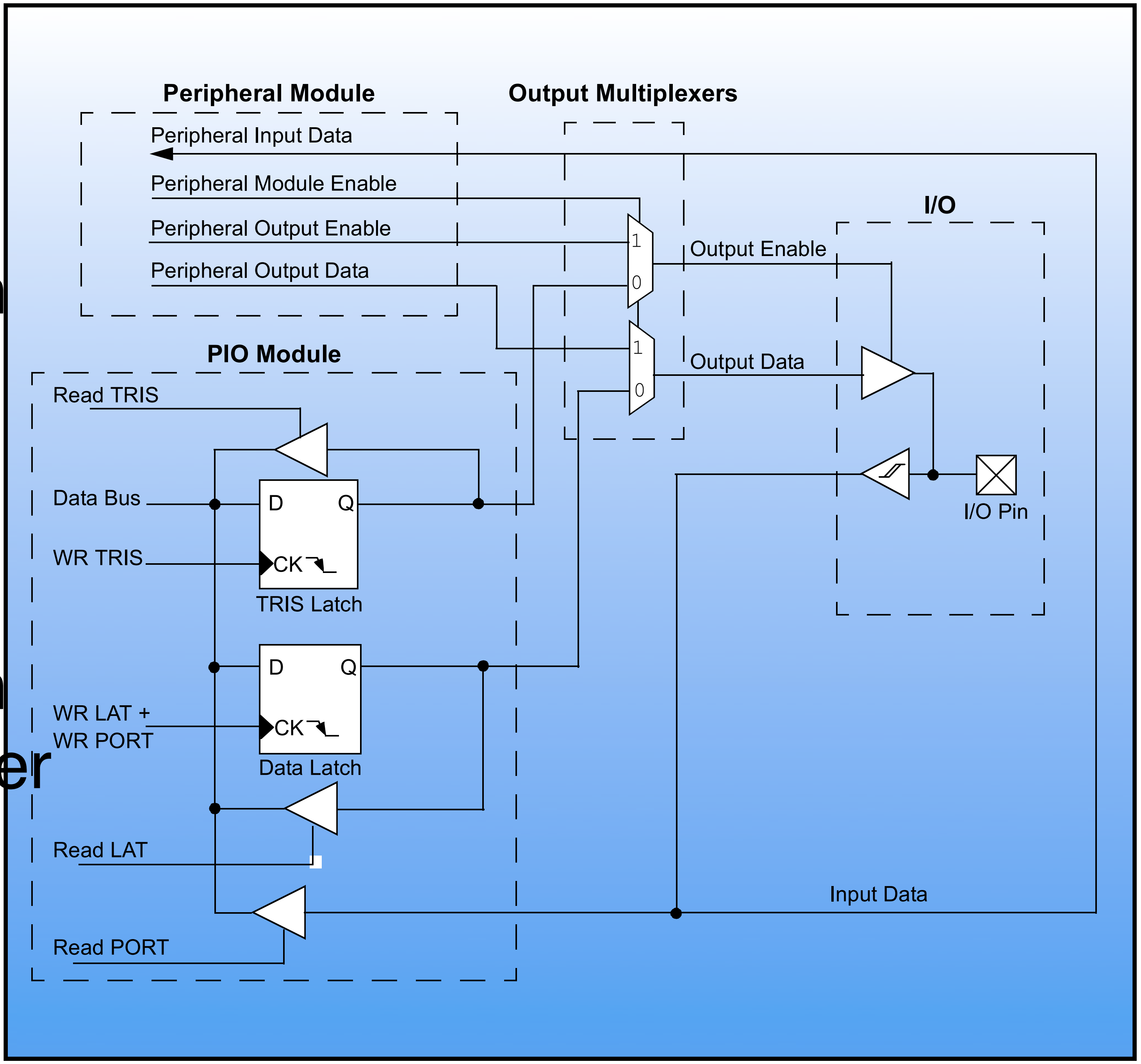
- Dual threshold
  - Once it switches state, another threshold applies to switch back to the original state
- Good for debouncing
  - bounce state => within threshold





I/O port schematic for the PIC24

- Output enable
  - puts value of latch onto the pin (pad)
- Input
  - reads from the pin after Schmitt trigger (dual threshold)



Software Debouncing

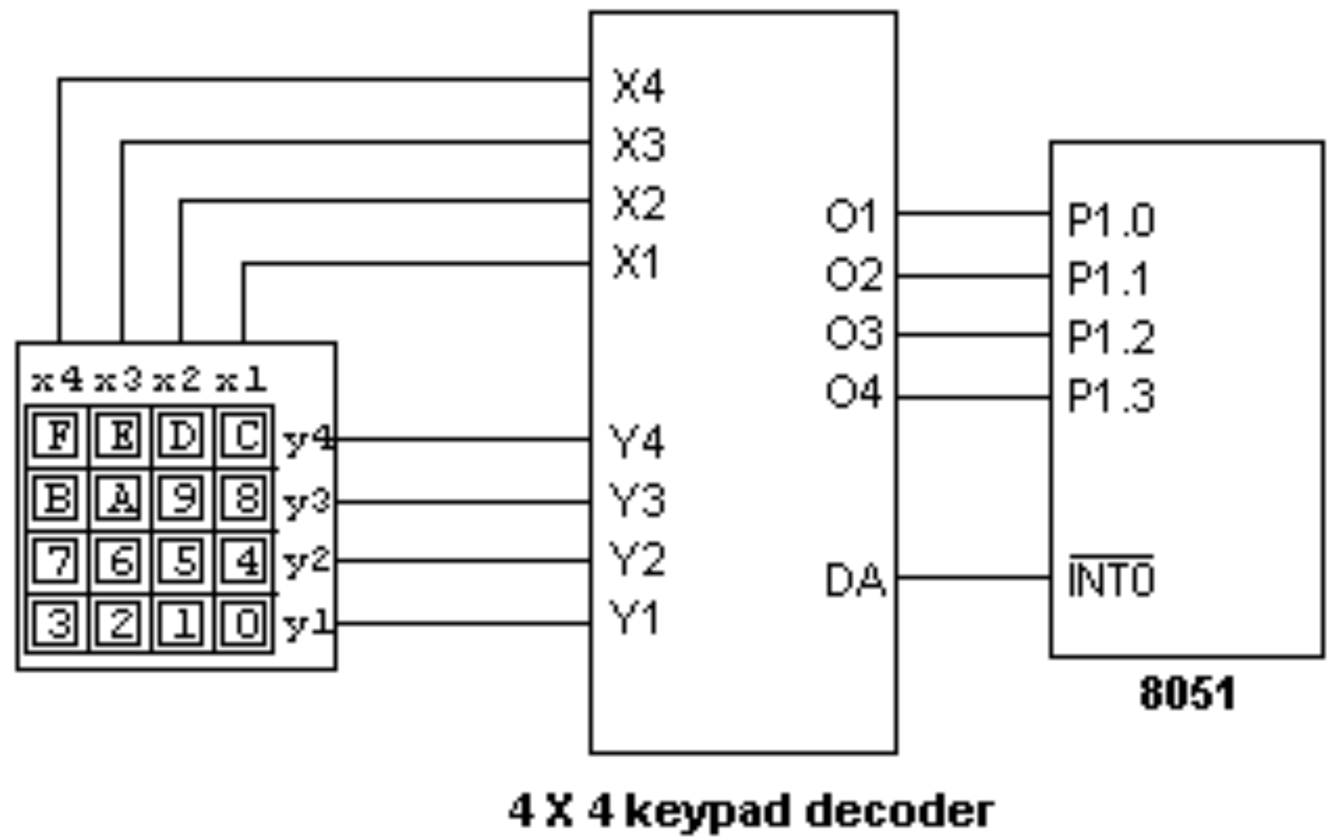
- Just delay for 10-30ms before checking the switch again
- Advantage: No need for extra hardware
- Disadvantage: busy loop can be wasteful
- Solution: use timer interrupt (later)

Keypad interfacing

- Keypad = matrix of push buttons
- Interface Option 1: one bit per key
  - +: enables 2^n combinations of key presses
  - -: needs n I/O pins => not scalable! (computer keyboards => 101 keys)
- Option 2: row/column scanning
  - Grows by sqrt(n) * 2

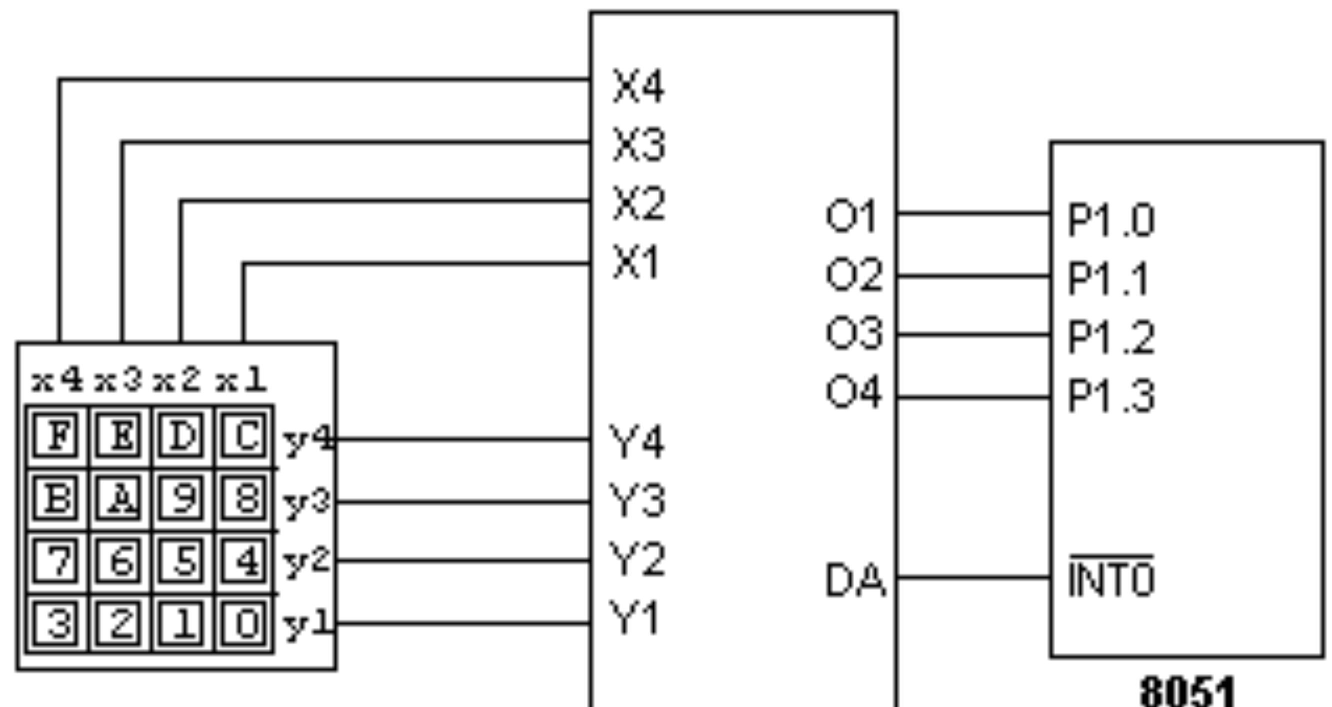
Hardware "decoder" for Keypad

- Encodes 16 keys using 4 bits to the MCU
- Handles debouncing
- DA (data available)
  - Goes low when any key pressed
  - Goes high when all released



Truth table for Decoder

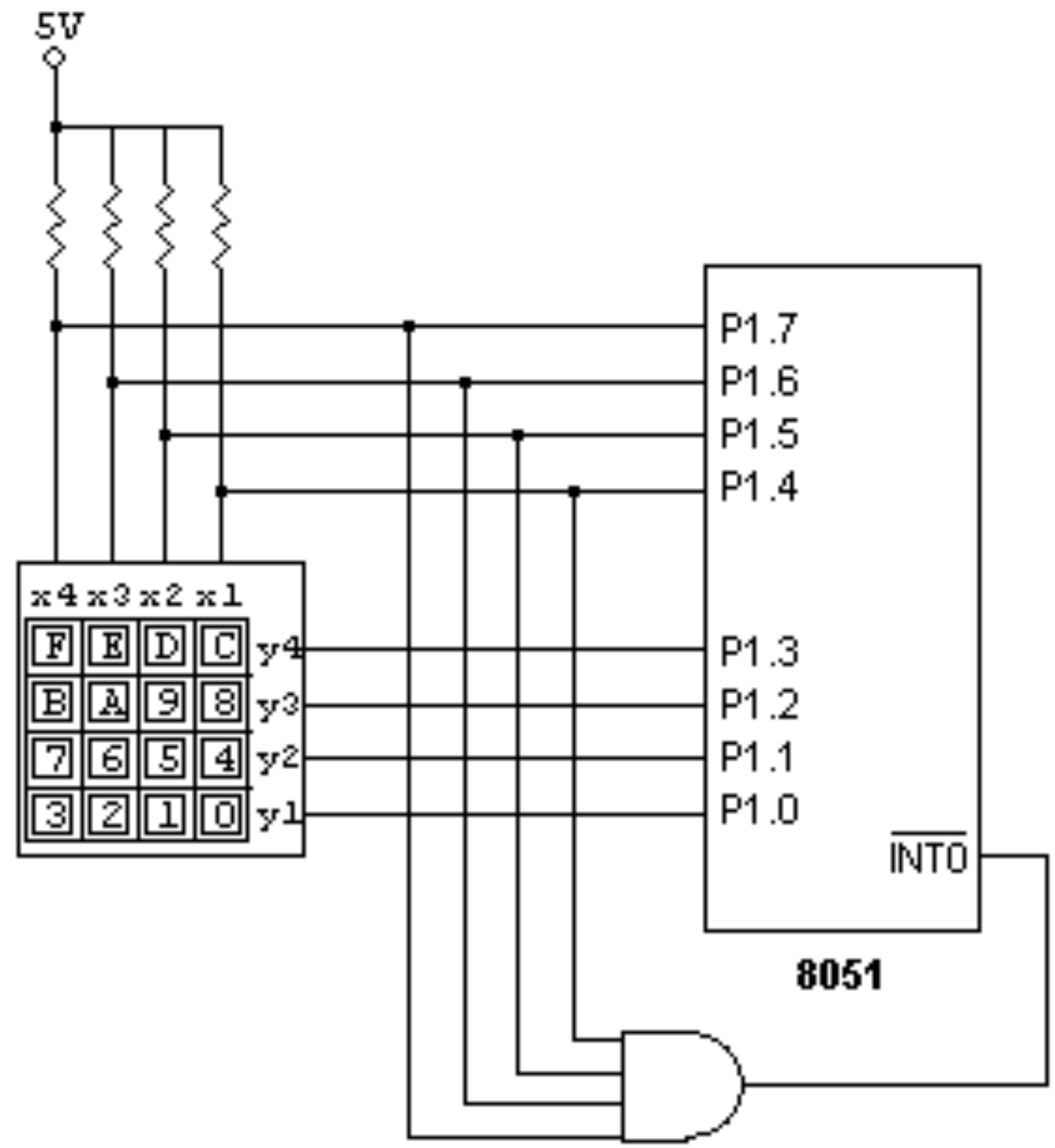
X:Y	O<4:1>
0001 0001	0000
0001 0010	0001
0001 0100	0010
0001 1000	0011
0010 0001	0100
0010 0010	0101
0010 0100	0110
0010 1000	0111
0100 0001	1000
0100 0010	1001
0100 0100	1010
0100 1000	1011
1000 0001	1100
1000 0010	1101
1000 0100	1110
1000 1000	1111



4 X 4 keypad decoder

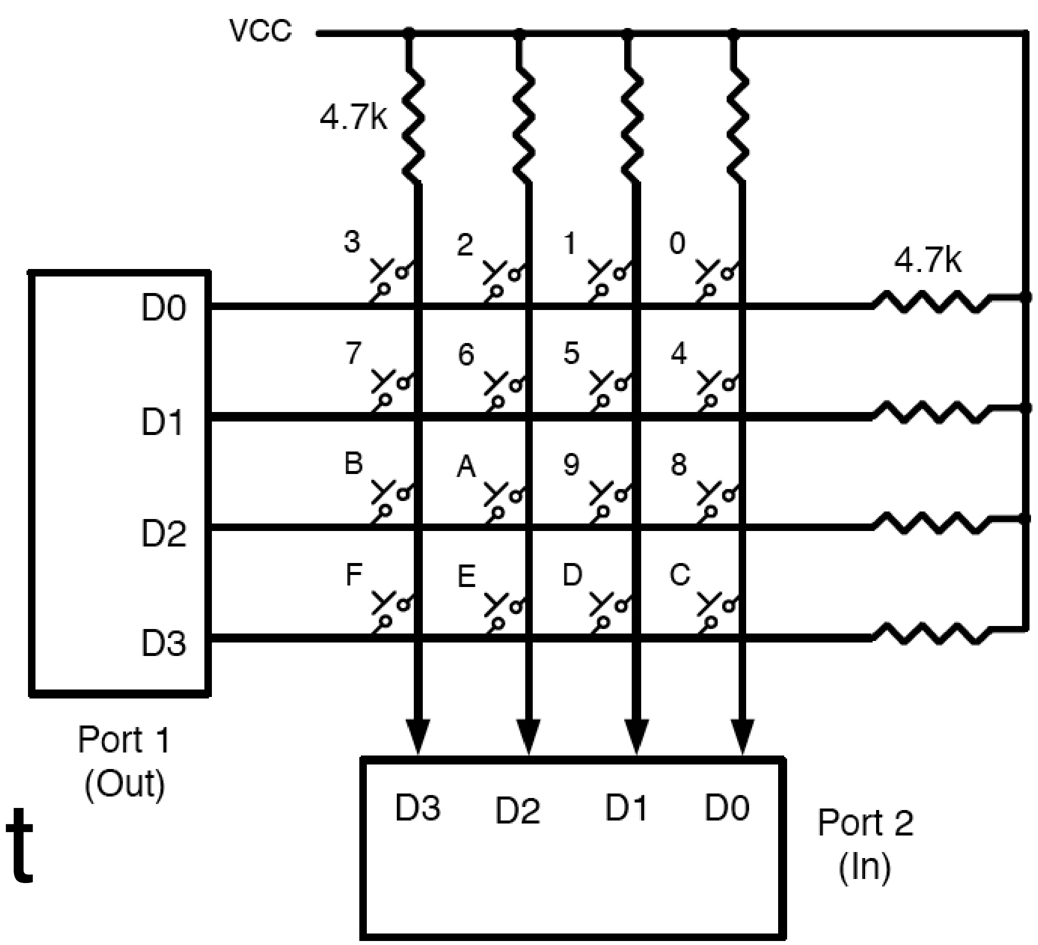
Software Decoder

- Scan one row at a time by setting that row to 0
  - Scan each column
    if 0, then pressed
    if 1, then not pressed
- DA line (optional) indicates if ANY key is pressed.



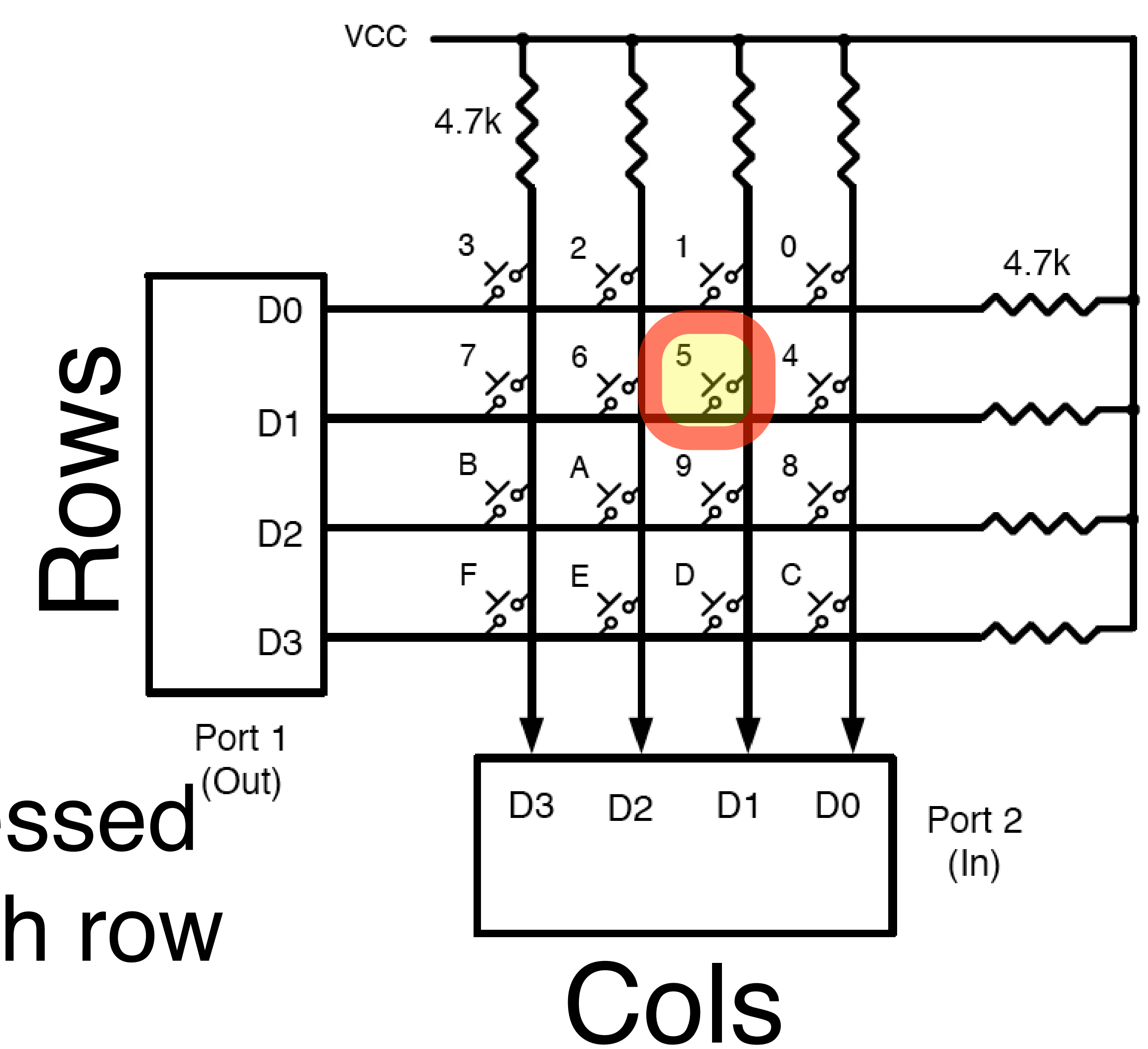
4x4 key matrix

- Pull-up lines
  - Press => connects Row & Column
- Connection
  - Rows: MCU output
  - Columns: MCU input



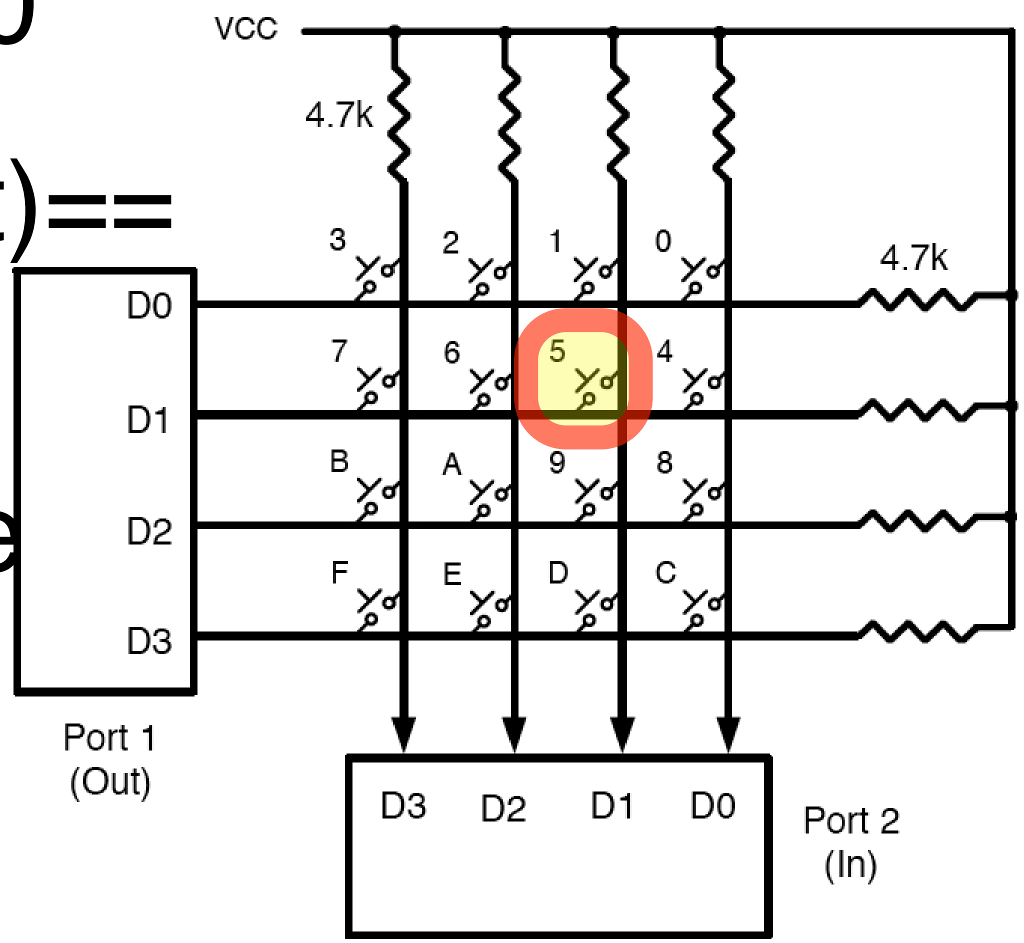
Key-press detection

- MCU sets Rows(output)=0000
- If no press, Cols(input)==1111
- If (key 5 is pressed), Cols(input) ==1101
  - Still need to find which Row(s) got pressed => serially check each row



Serially checking each row (for given col)

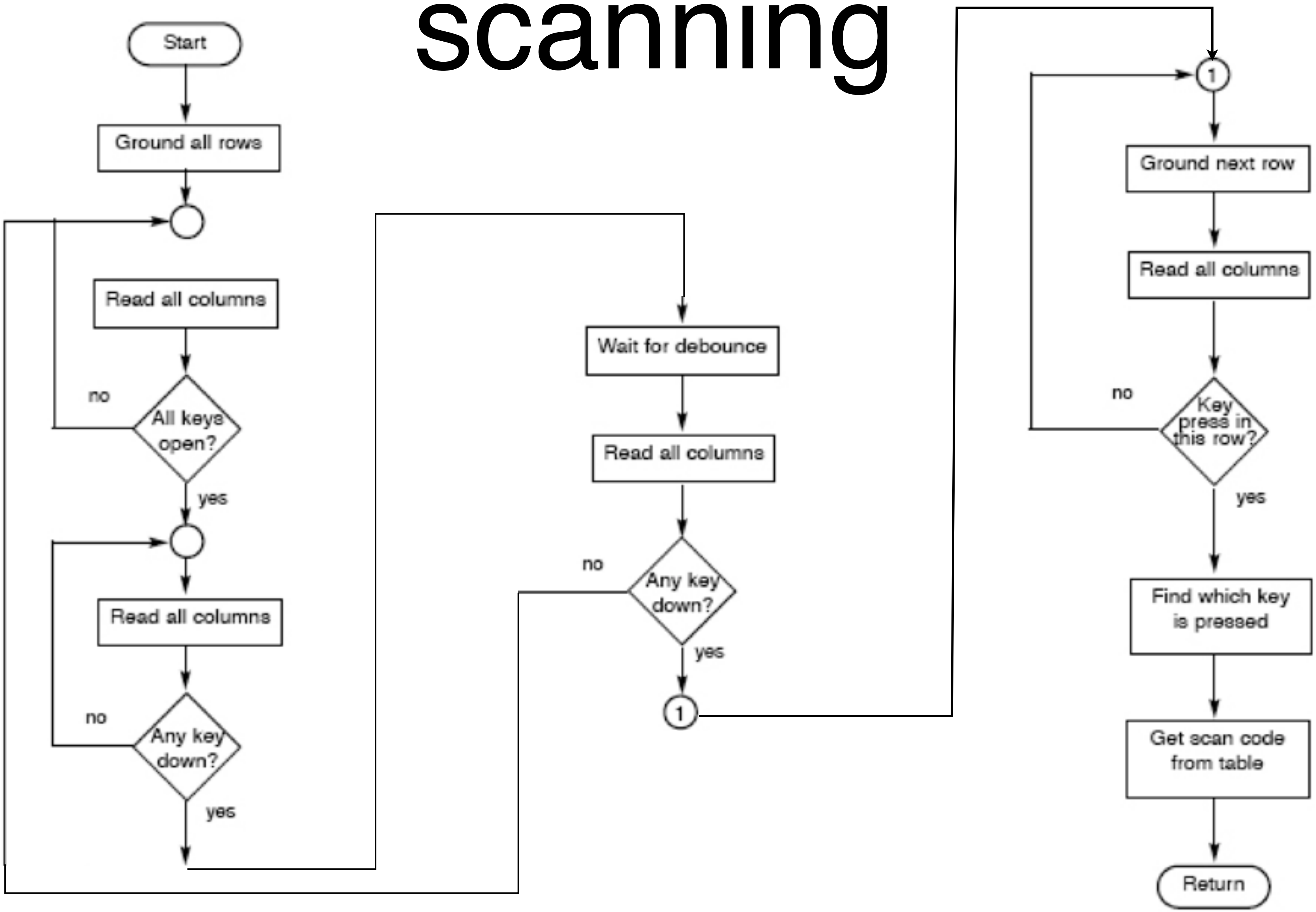
- MCU sets Rows(output)= 0111, 1011, 1101, 1110
- MCU reads Cols(input)== 1111, 1101, 1111, 1111
- 0s yield the coordinate of the pressed button => multi-button press can be detected too!



Software debouncing for Keypad

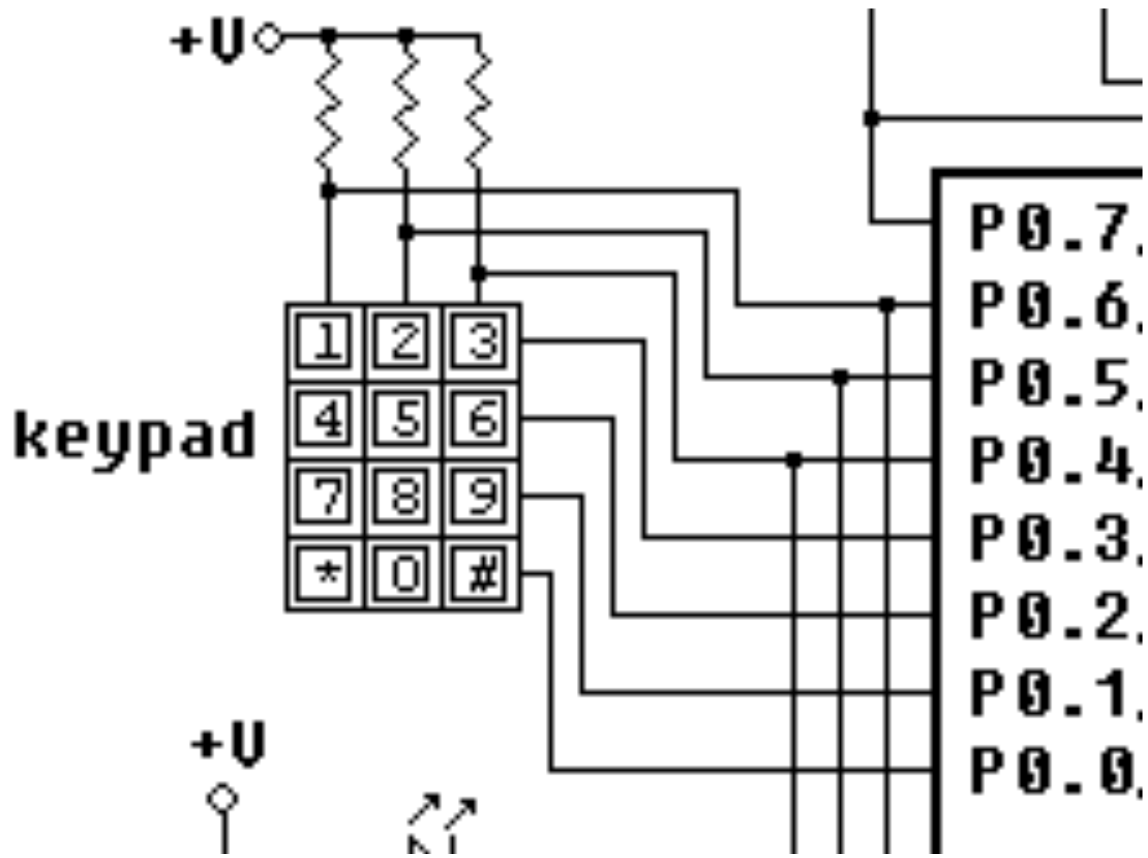
- you can still do software debouncing
- Option 1: check DA line like a button
- Option 2: while scanning row/col
  - start busy waiting as soon as you find one
  - Issue: compatible with multiple keys?

Flowchart for keypad scanning



Keypad in EdSim51

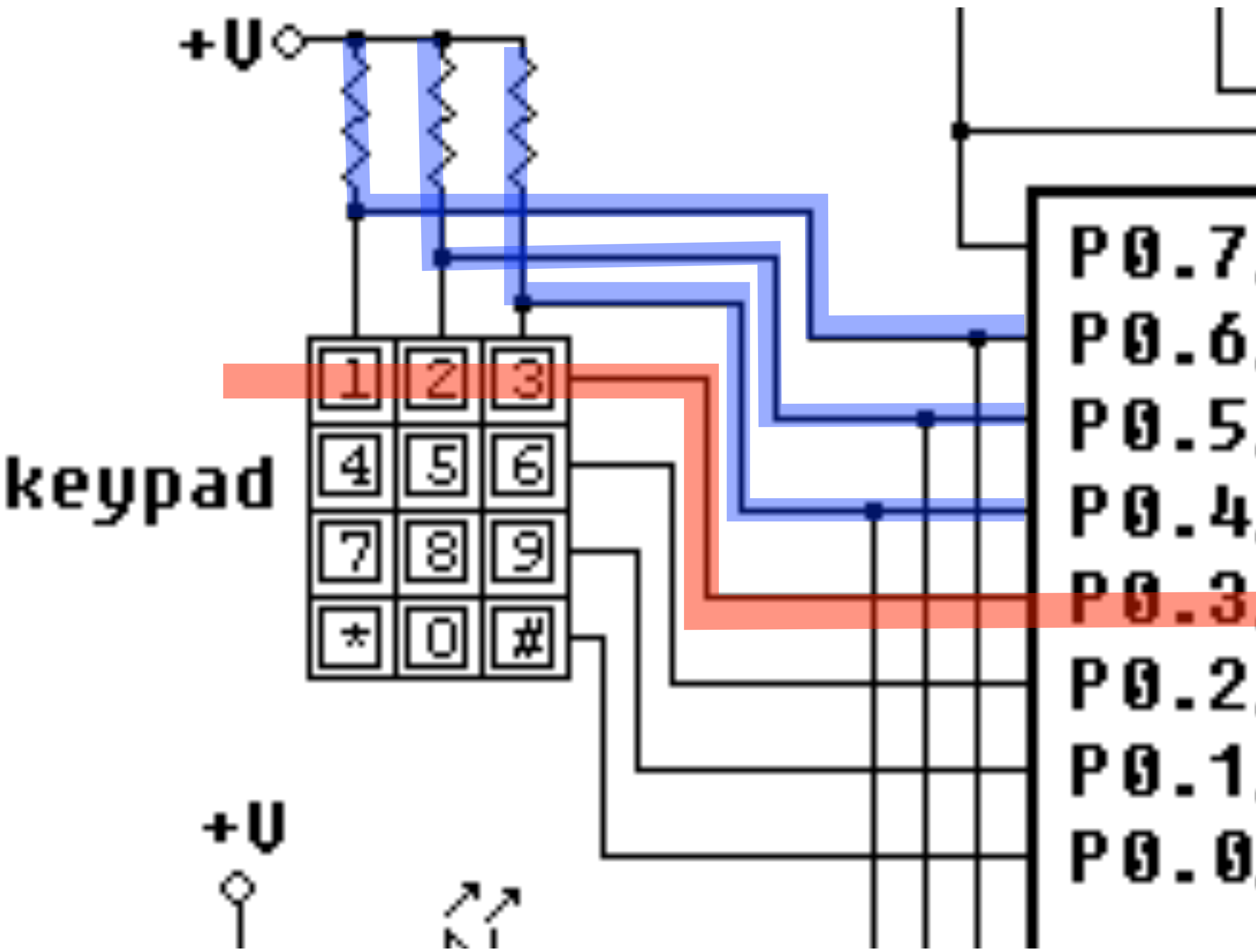
- Set P0.4-P0.6 to 111
    - to enable input of columns
- Take turn setting P0.0-P0.3 to 0 one bit at a time
    - to select a row



scankey.c

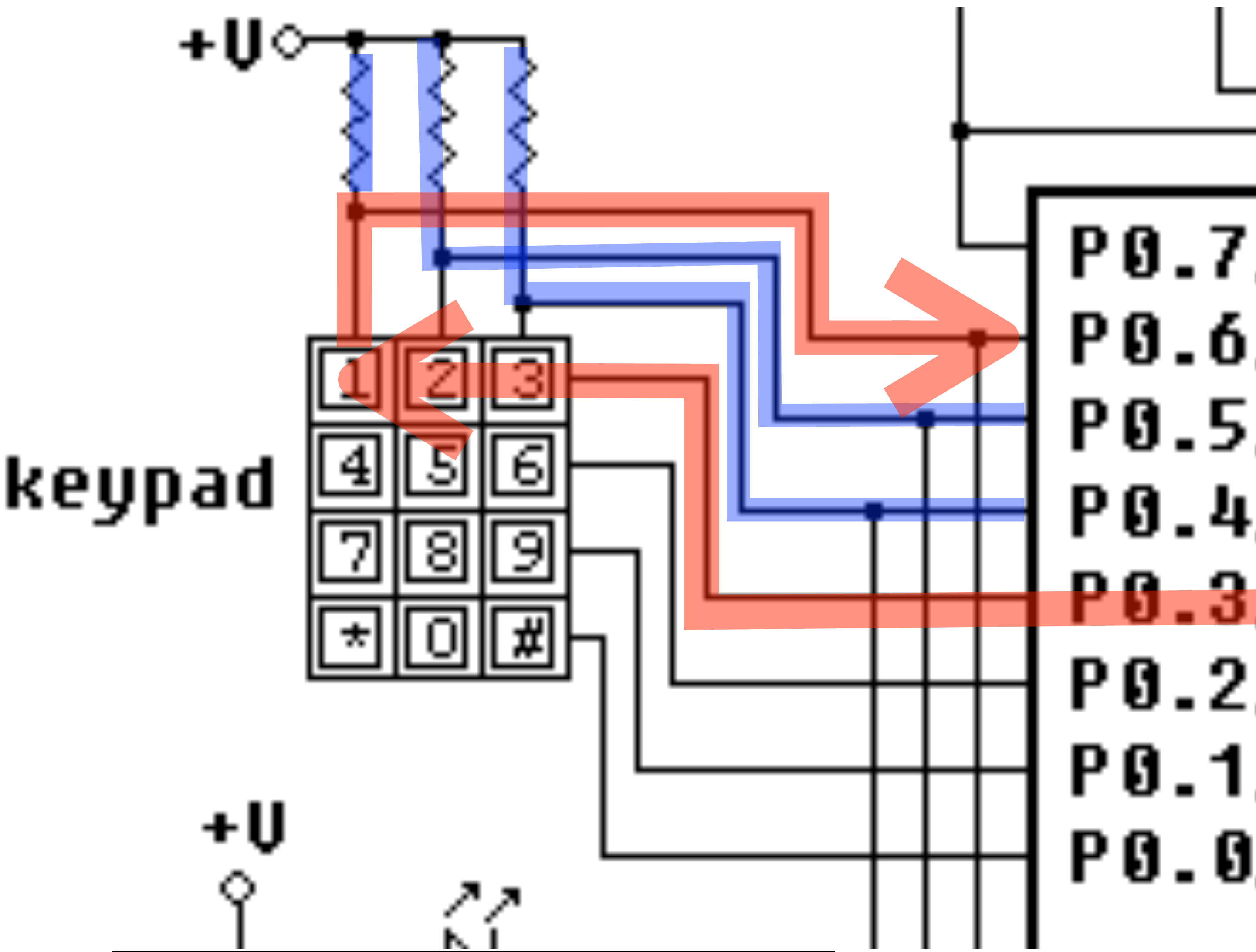
```
#include <8051.h>
char colScan(char c);
void Main(void)  {
    char bitmap, bitmapL, bitmapH;  char row, rowmask;
    while (1) {
        for (row=bitmapL=bitmapH=0, rowmask = 0xf7; row < 4;  row++, rowmask >>= 1) {
                bitmap=colScan(0xFE);
                if (row==2) {
                    bitmapH = (bitmapL >> 2);
                }
                bitmapL = (bitmapL<<3) | bitmap;
        }
        P1 = bitmapL;
        P2 = bitmapH;
      }
}
char colScan(char rowmask) {
        P0 = rowmask;
        return (~(P0>>4)) & 0x07;
}
```

To test row 0 (keys 1,2,3) when no key pressed



pin	direction	port latch
-	-	-
1	input	1
1	input	1
1	input	1
0	output	0
1	output	1
1	output	1
1	output	1

Testing row 0 while key [1] pressed



pin	direction	port latch
-	-	-
0	input	1
1	input	1
1	input	1
0	output	0
1	output	1
1	output	1
1	output	1

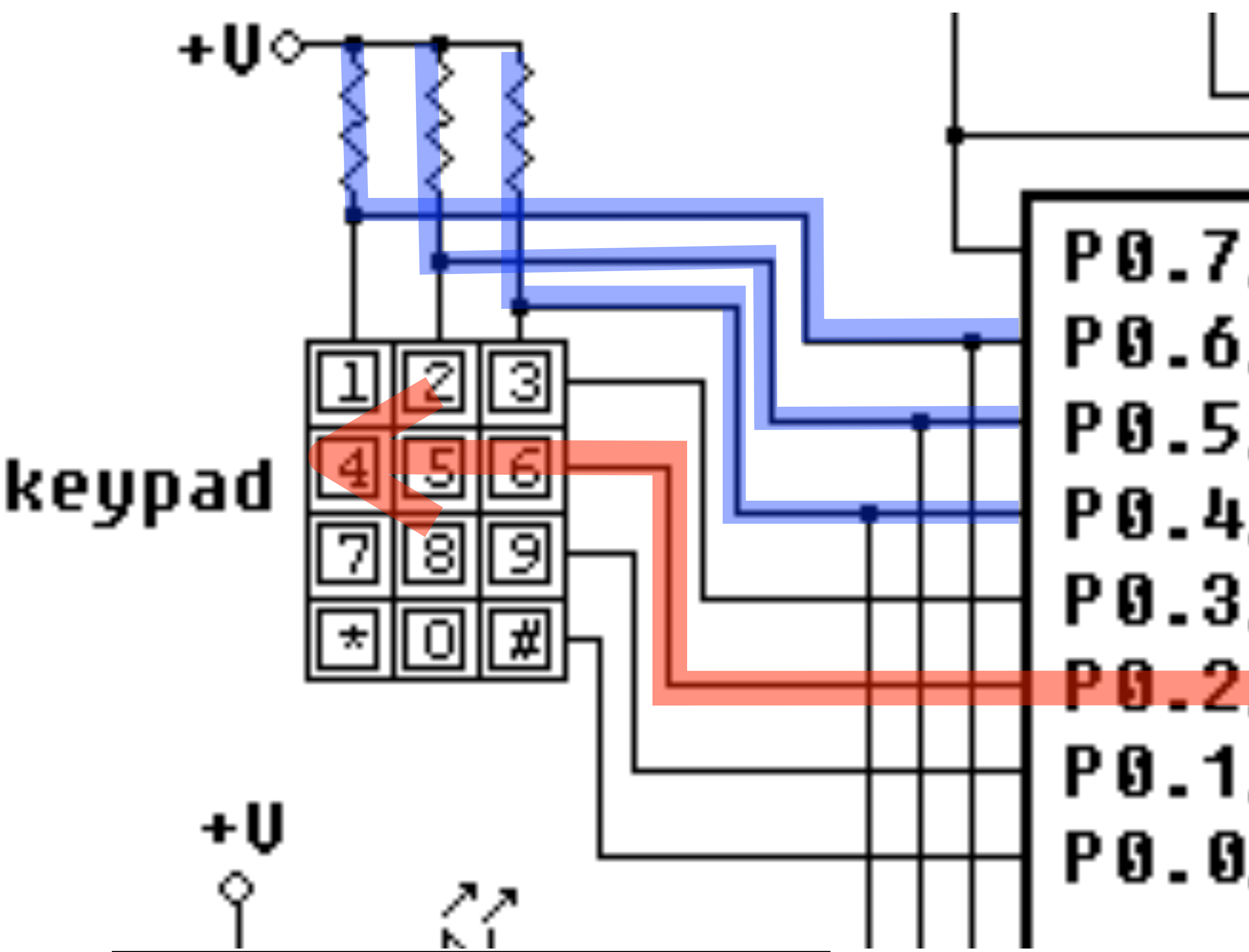
main loop in scankey.c

bitmapH								bitmapL							
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
													1	2	3

rowmask							
1	1	1	1	0	1	1	1

to write to P0

Testing row 1 while key [1] pressed



pin	direction	port latch
-	-	-
1	input	1
1	input	1
1	input	1
1	output	1
0	output	0
1	output	1
1	output	1

main loop in scankey.c

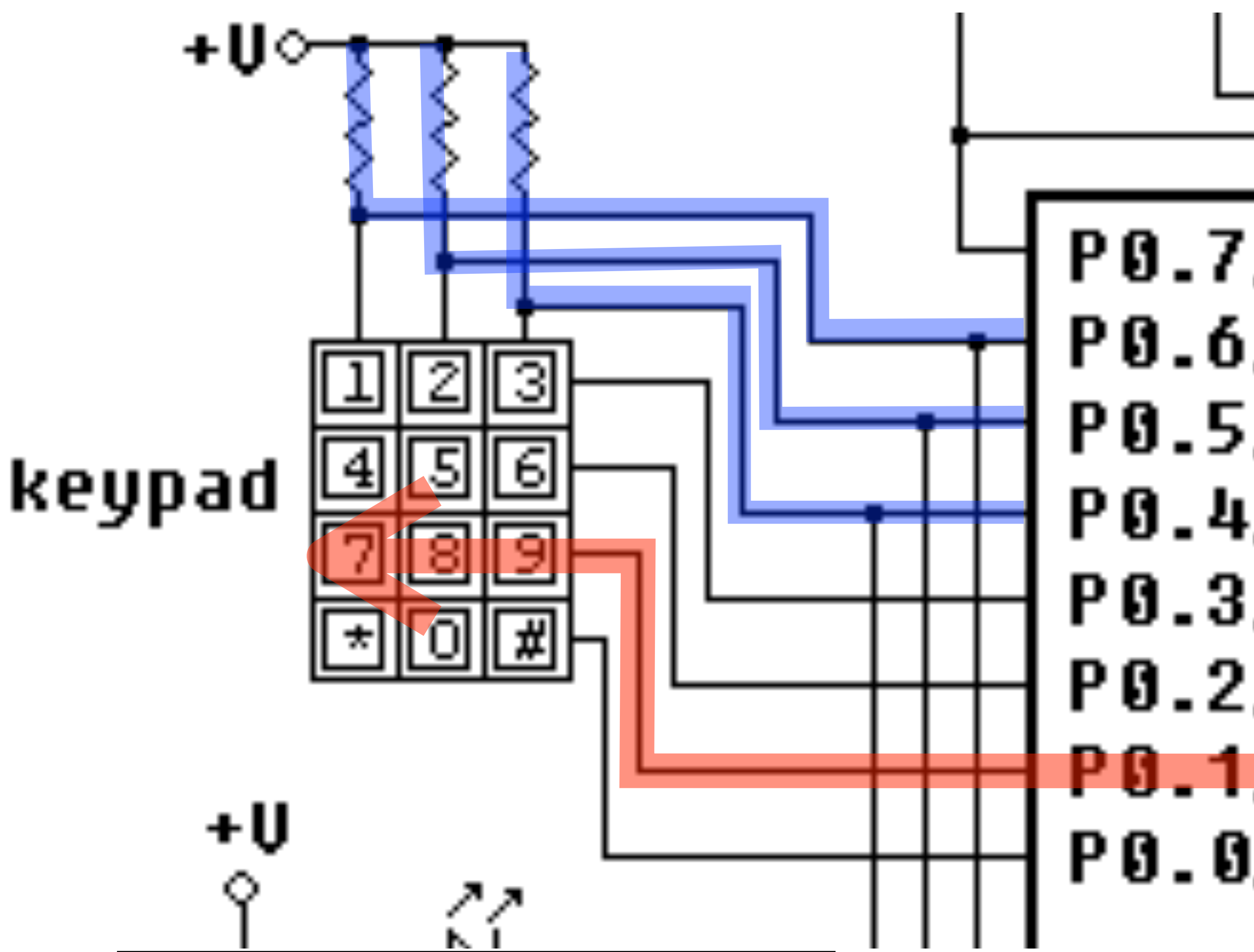
bitmapH								bitmapL							
0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0
										1	2	3	4	5	6

bitmapL = (bitmapL << 3) | colScan(...)

rowmask							
1	1	1	1	1	0	1	1

to write to P0

Testing row 2 while key [1] pressed



pin	direction	port latch
-	-	-
1	input	1
1	input	1
1	input	1
1	output	1
1	output	1
0	output	0
1	output	1

main loop in scankey.c

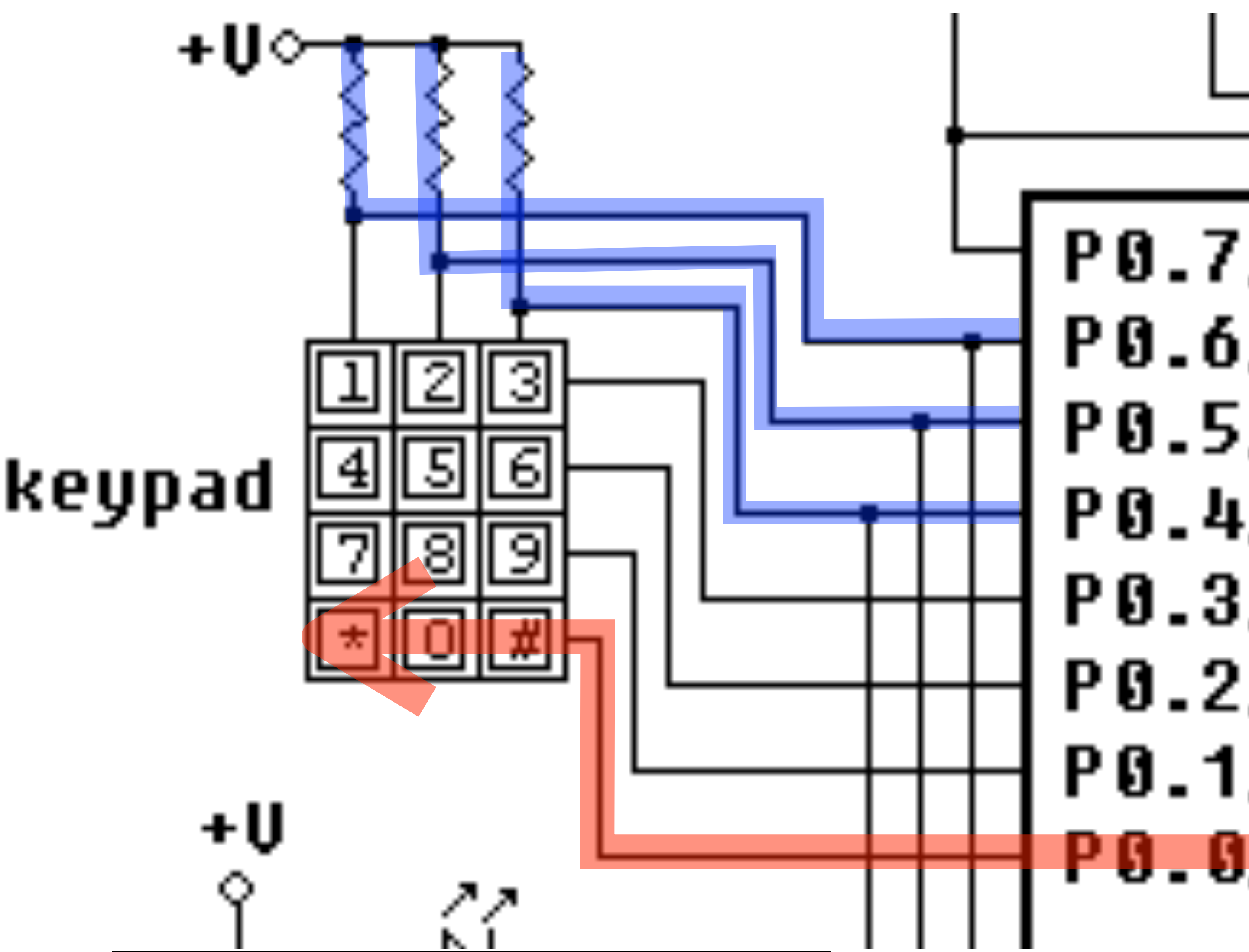
bitmapH								bitmapL							
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
				1	2	3	4	2	3	4	5	6	7	8	9

```
if (row==2) {
  bitmapH = (bitmapL>>2);
}
bitmapL = (bitmapL<<3) |
```

rowmask							
1	1	1	1	1	1	0	1

to write to P0

Testing row 3 while key [1] pressed



pin	direction	port latch
-	-	-
1	input	1
1	input	1
1	input	1
1	output	1
1	output	1
1	output	1
0	output	0

main loop in scankey.c

P2 = bitmapH								P1 = bitmapL							
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
				1	2	3	4	5	6	7	8	9	*	0	#

bitmapL = (bitmapL<<3) |

rowmask							
1	1	1	1	1	1	1	0

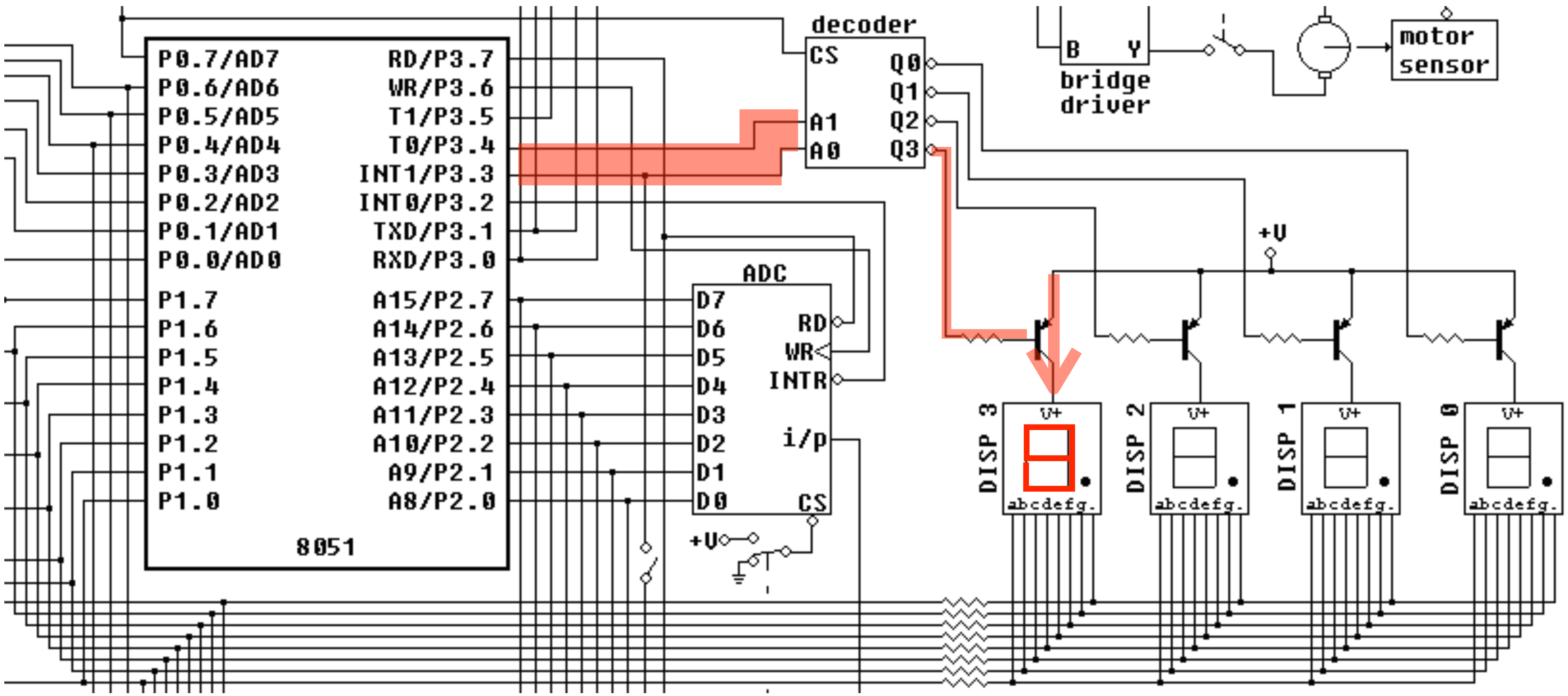
to write to P0

Applying row/col multiplexing to LED

- Can control LEDs by row-col scanning
- Hardware solution
  - Use a hardware decoder or demultiplexer
- Software solution
  - Use row-column multiplexing

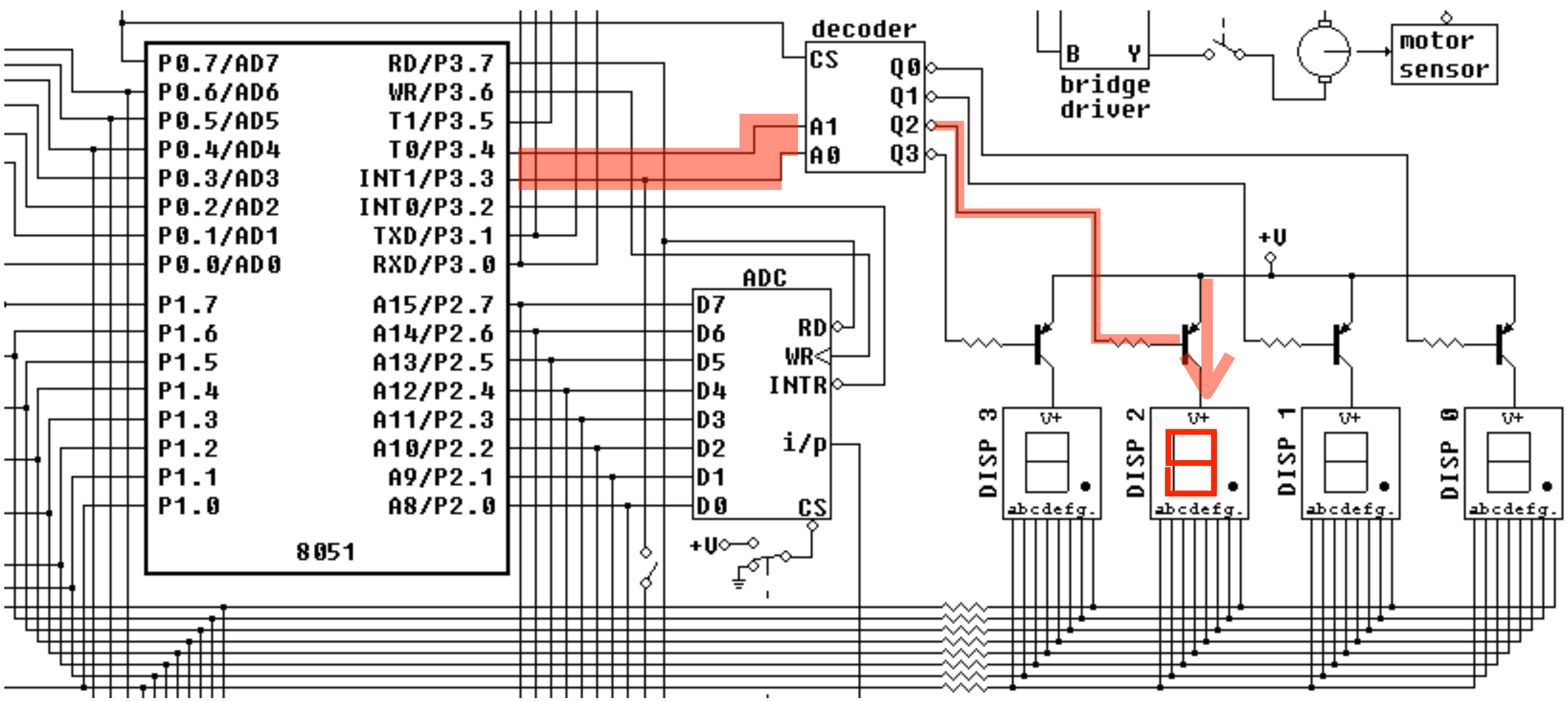
EdSim Example: 7-segment LED

A<1:0> = P3.<4:3> = 11, CS = 1



EdSim Example: 7-segment LED

A<1:0> = 10



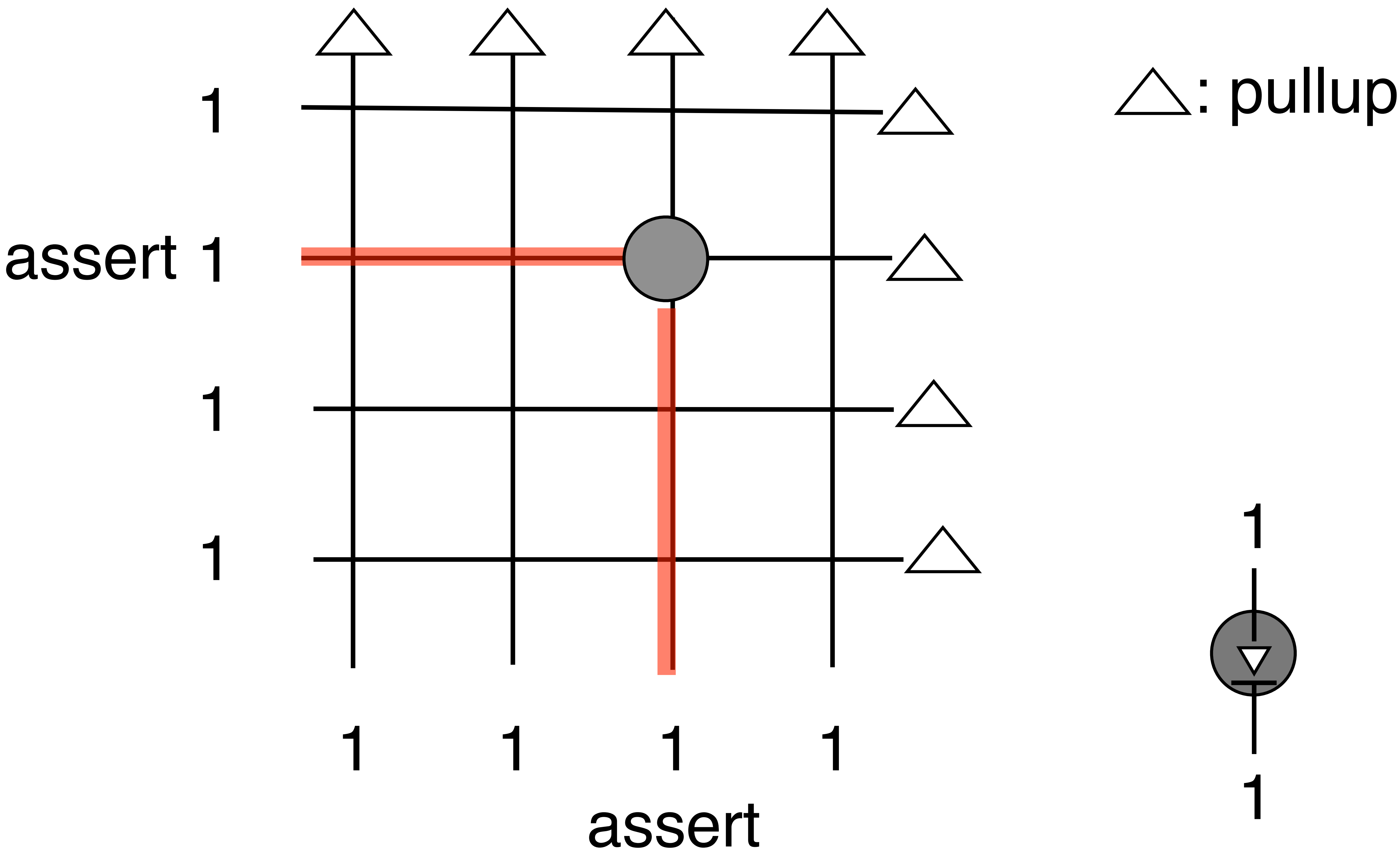
Rapidly cycling through all digits

- Digits could appear to flicker
  - If fast enough, then eyes might not see flicker
- Frequency above flicker perception
  - 60 Hz (some can tell)
  - $\geq$ 200 Hz -- most people can't tell.

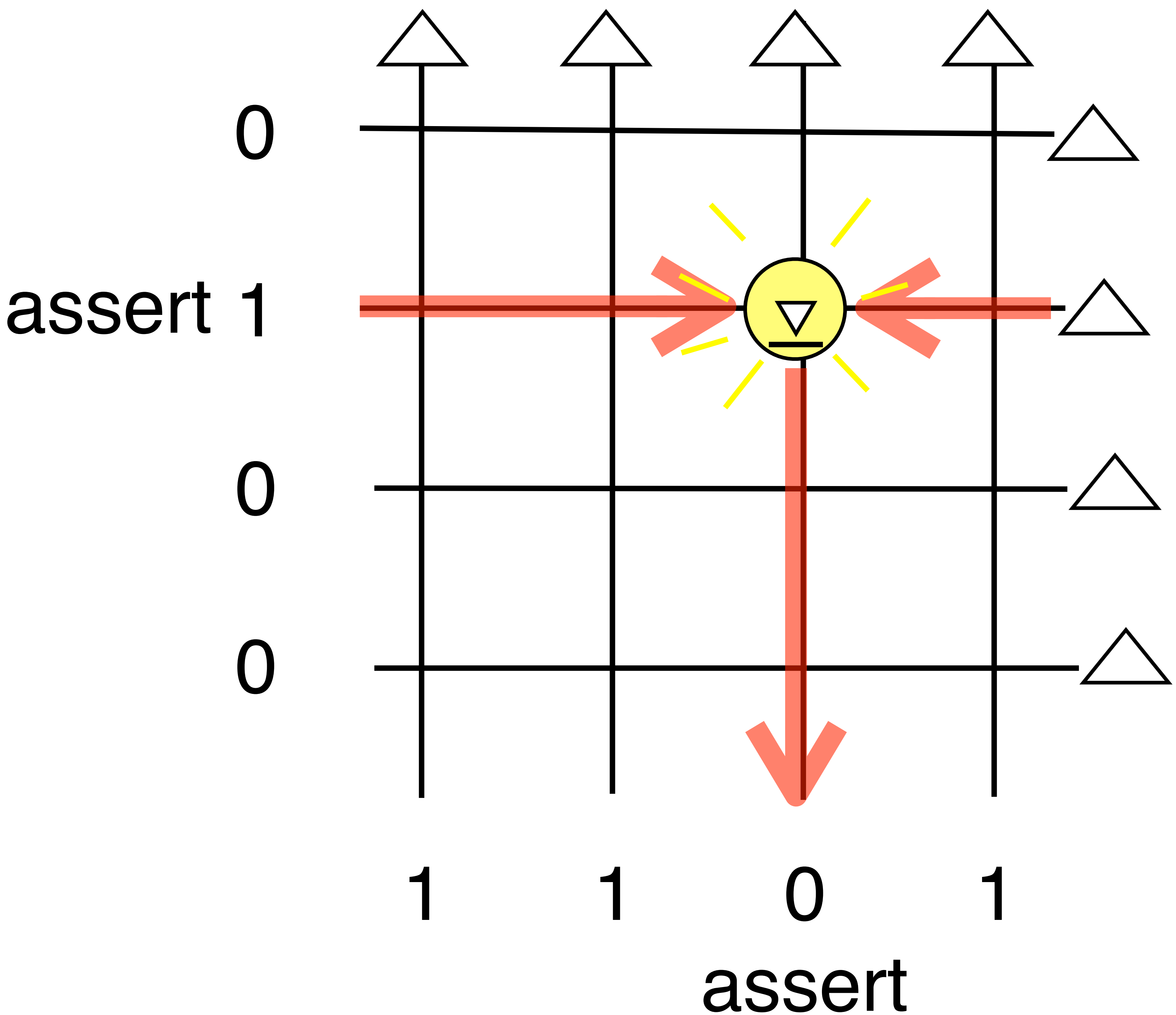
Software solution

- Analogous to row-column scanning
  - Difference: Row-column both output
- Connect LED from Row to Col
  - Row = 1, Col = 0 to turn LED on
  - Row = 0, Col = 1 to keep LED off

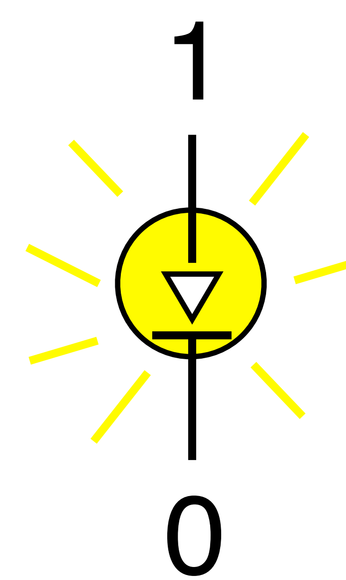
Normally, LED off



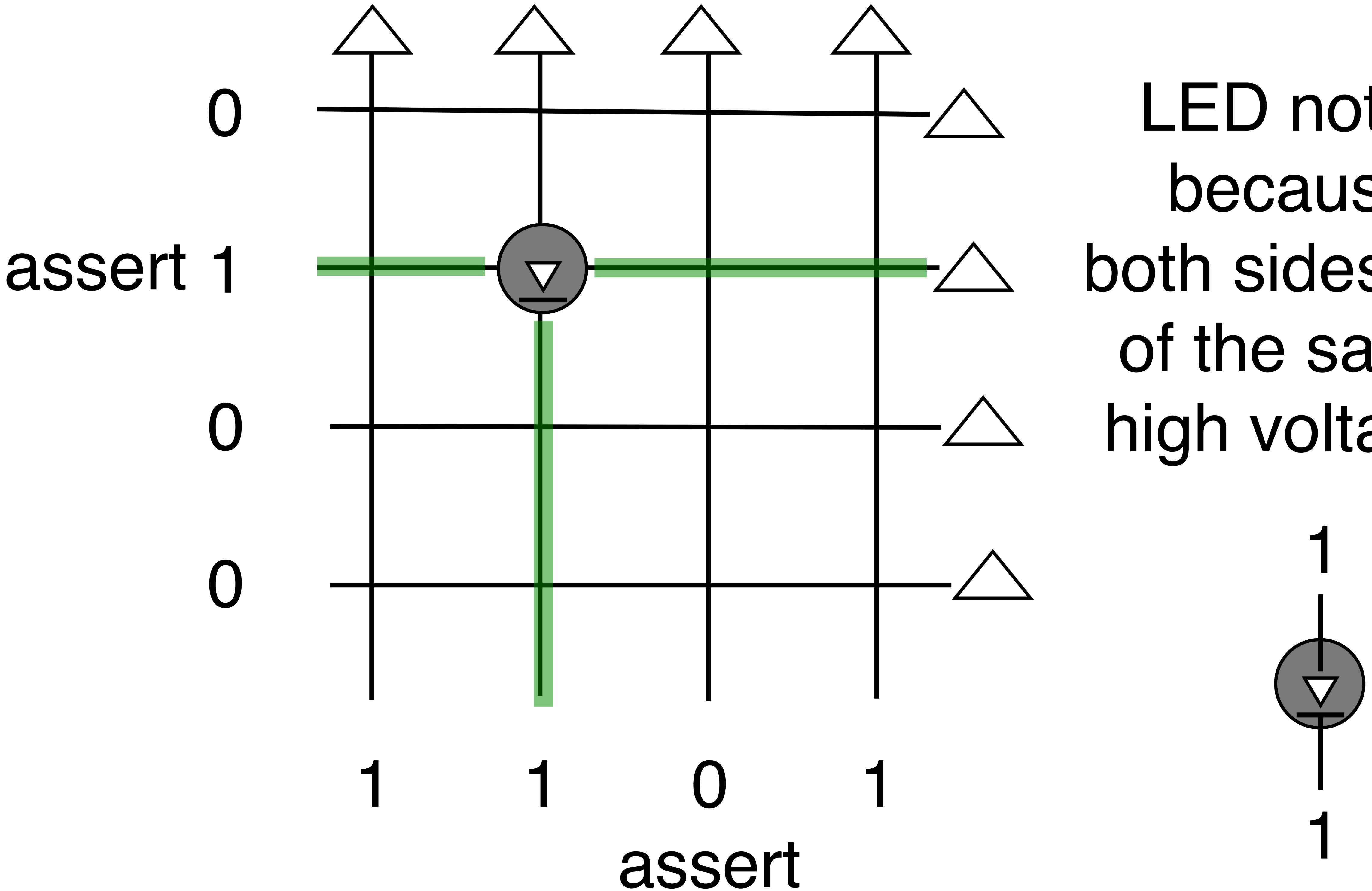
LED Select



Loop the rows & columns to turn on one LED at a time

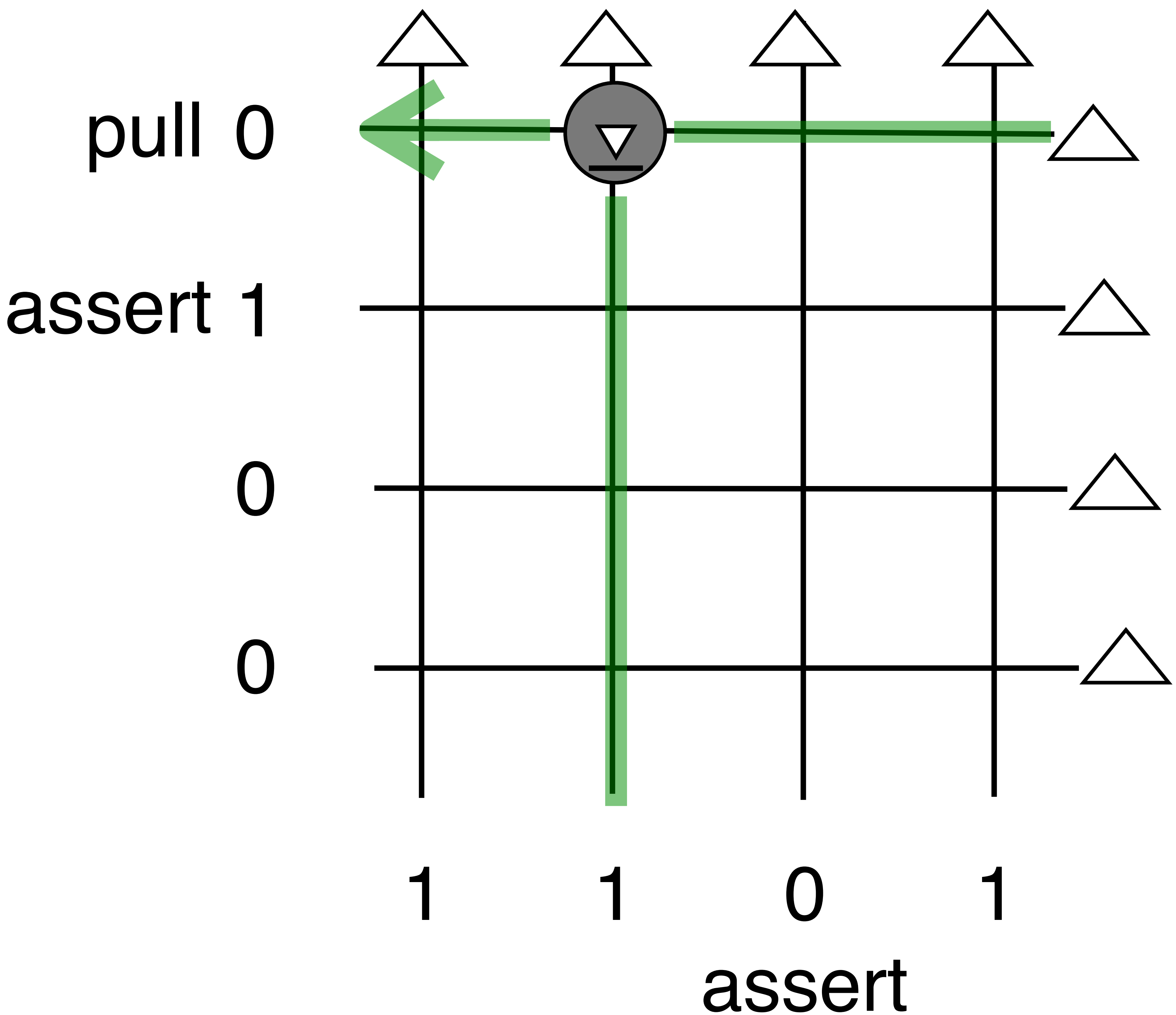


LED unselected: case 1

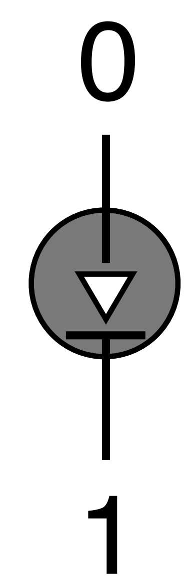


LED not lit because both sides are of the same high voltage!

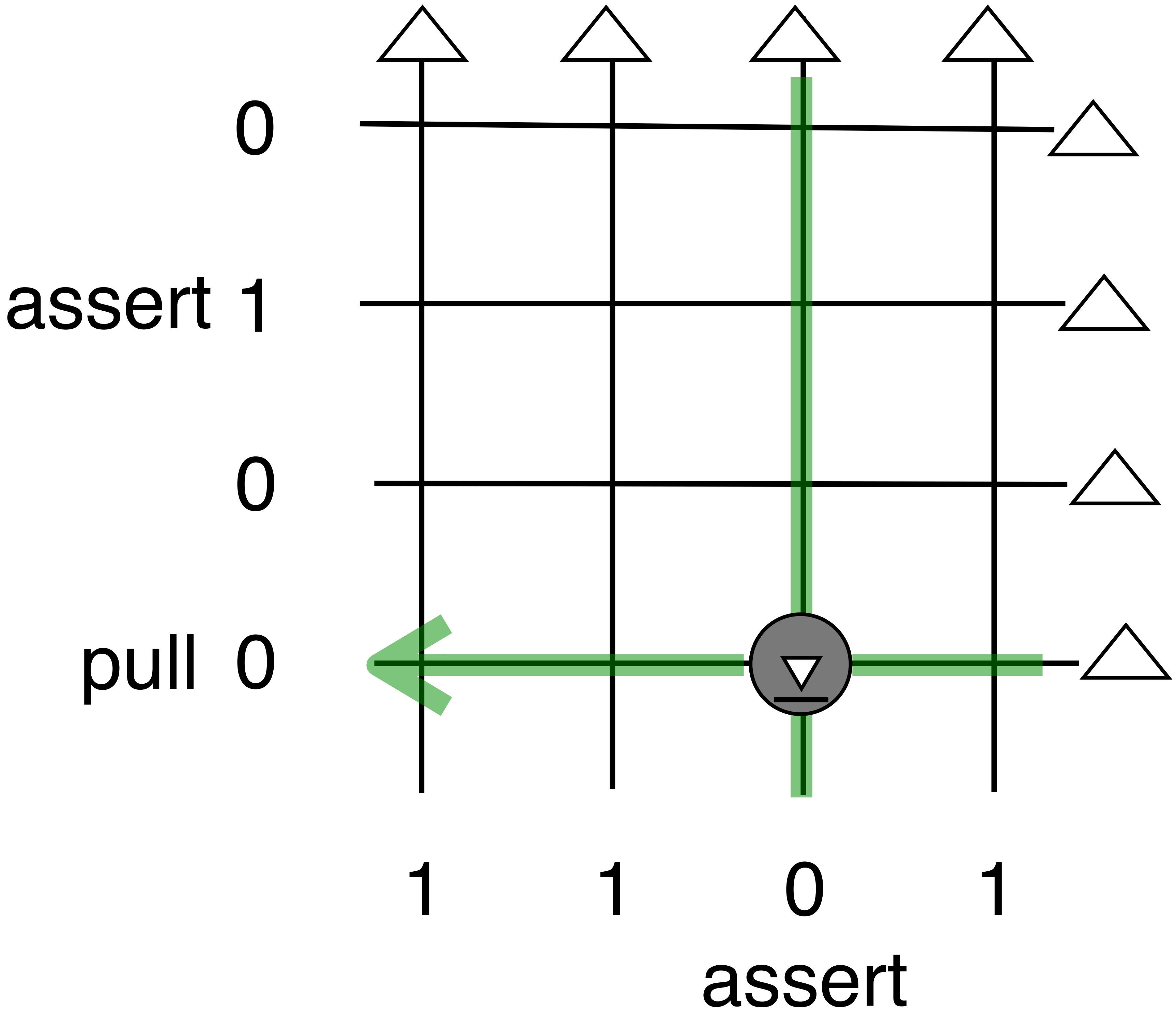
LED unselected: case 2



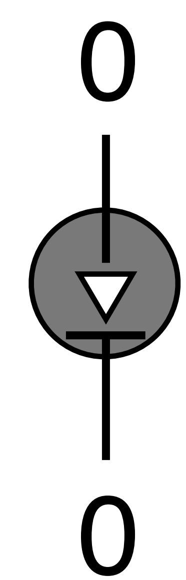
LED not lit because polarity is reversed!



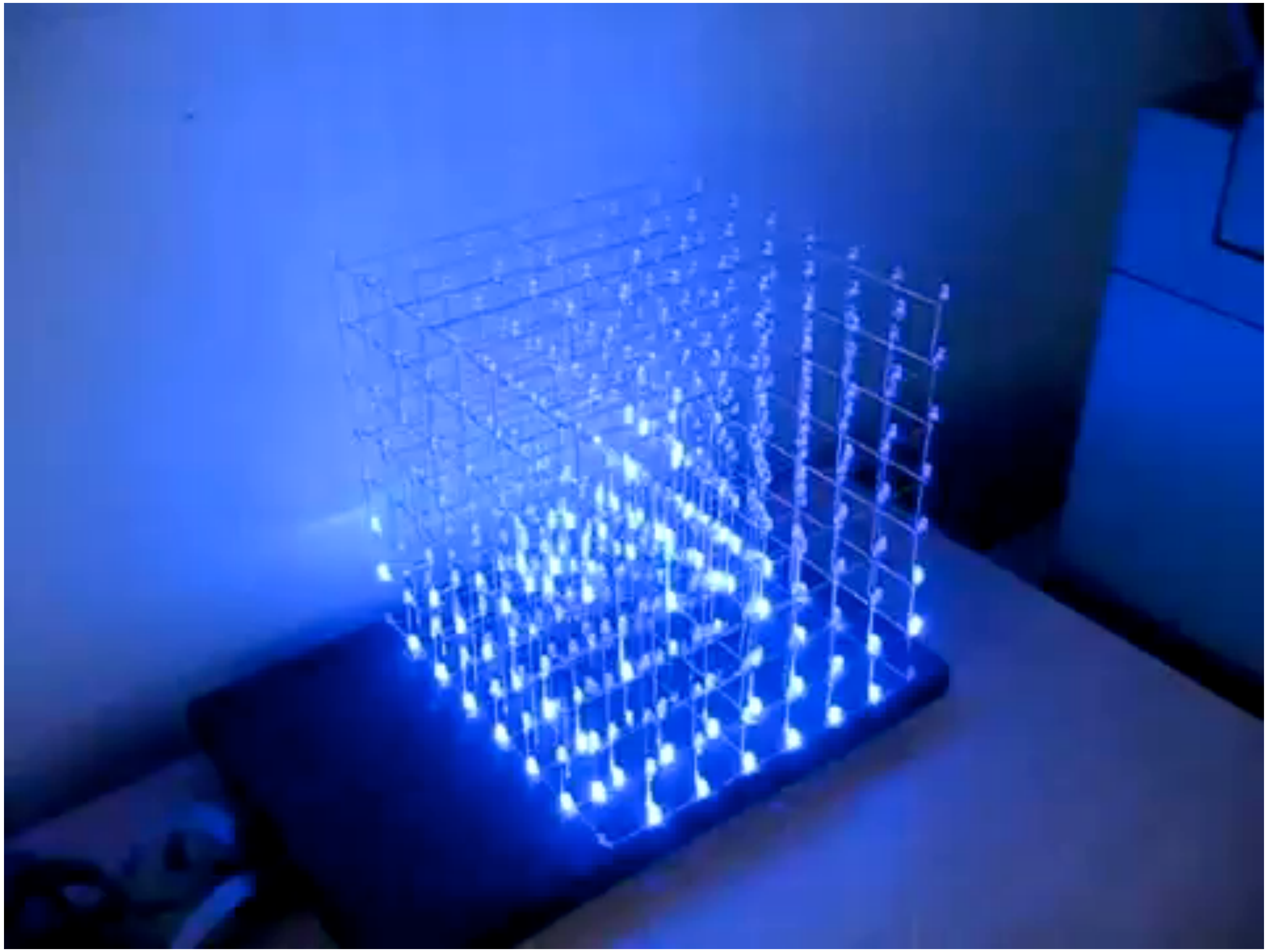
LED unselected: case 3



LED not lit because both sides are of same low voltage



Application of Row/Col: 3D Cube Light

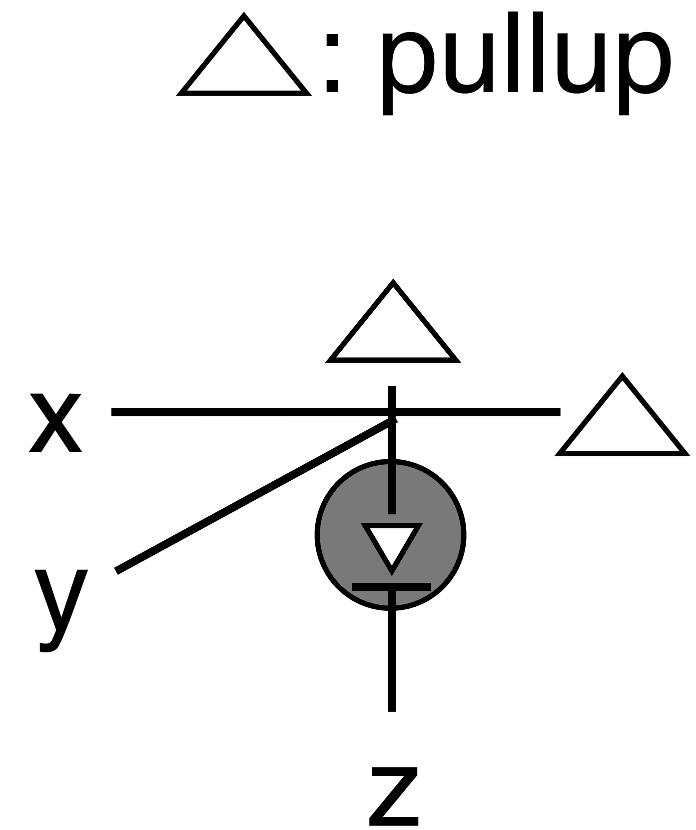


Number of GPIO pins needed

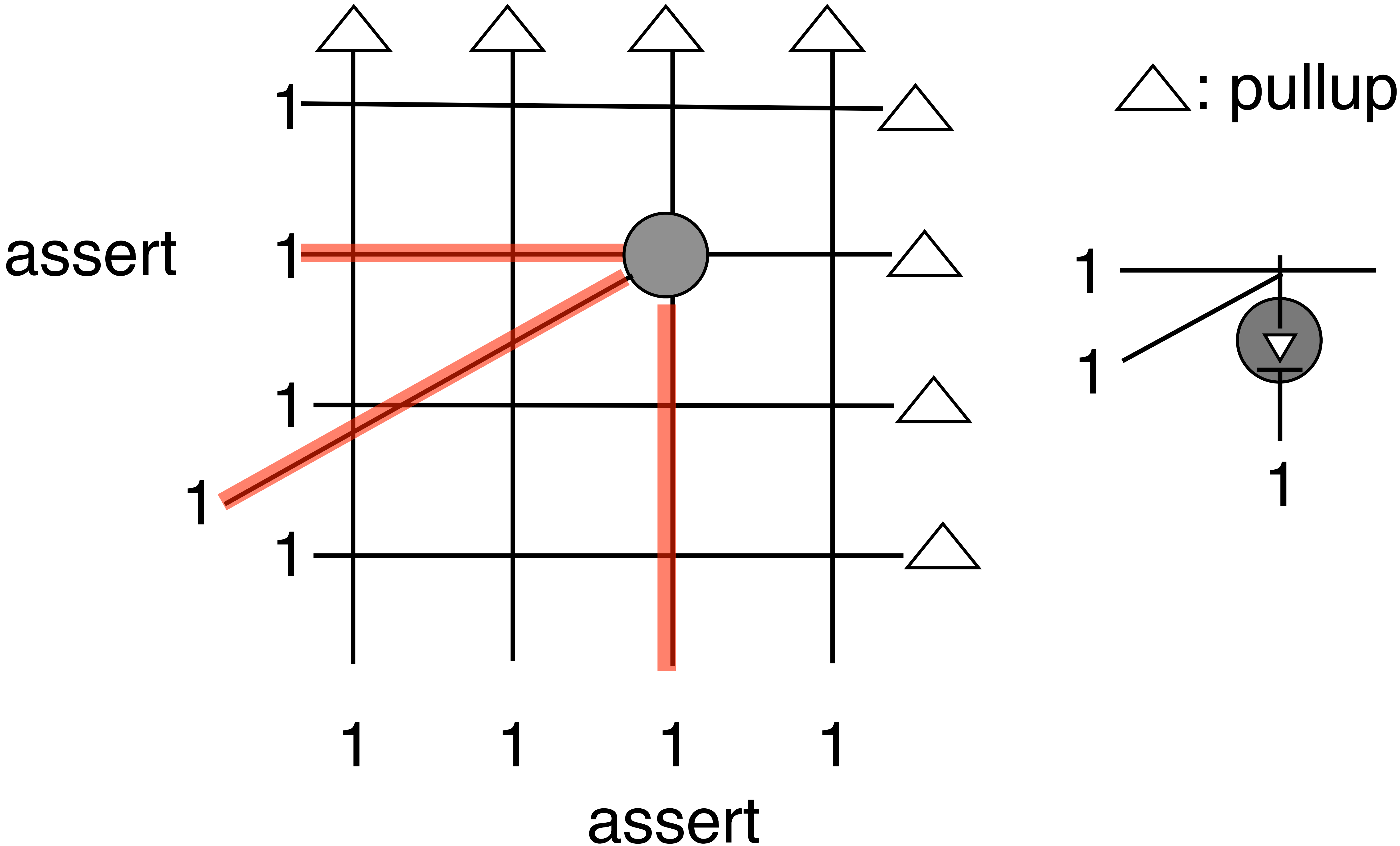
- Usually, 2 x square root of total pins
  - 8x8x8 cube: 2 x sqrt(512) = 2 x 23 = 46 pins
  - too many for original 8051 (32 GPIO)
- Alternative:
  - generalize schematic to 3D!!!

Generalizing to 3D

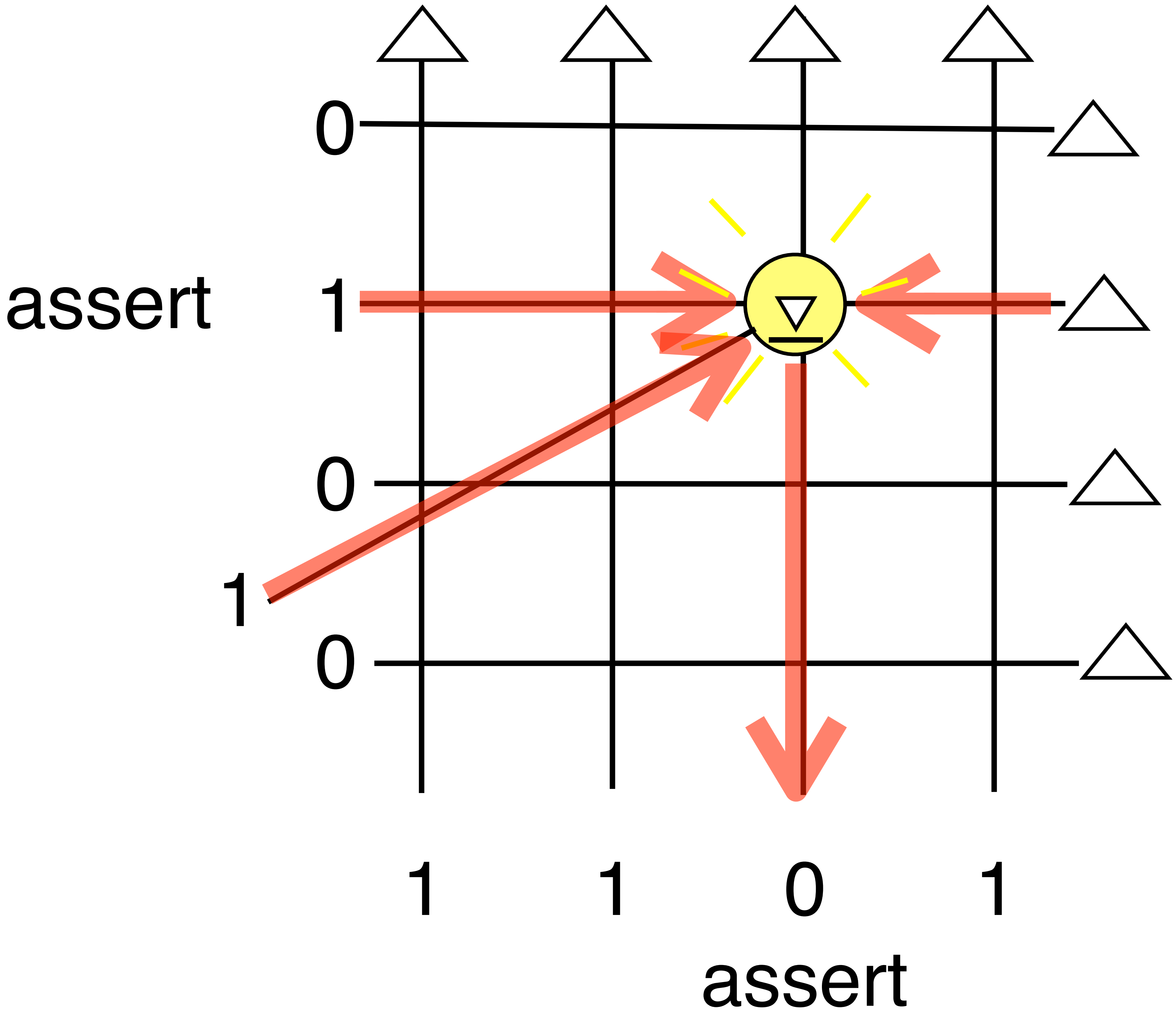
x	y	z	state
0	0	0	off
0	0	1	off
0	1	0	off
0	1	1	off
1	0	0	off
1	0	1	off
1	1	0	on
1	1	1	off



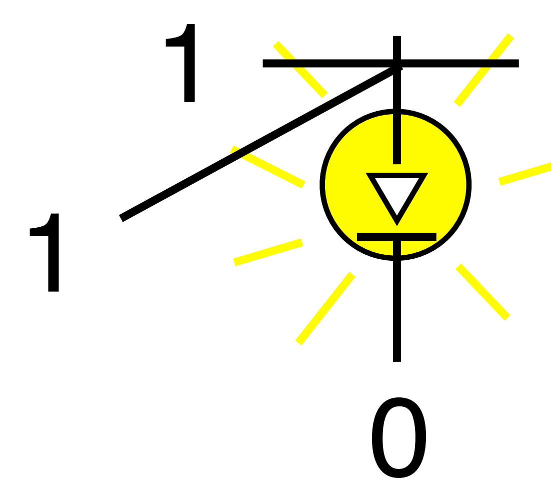
Normally, LED off



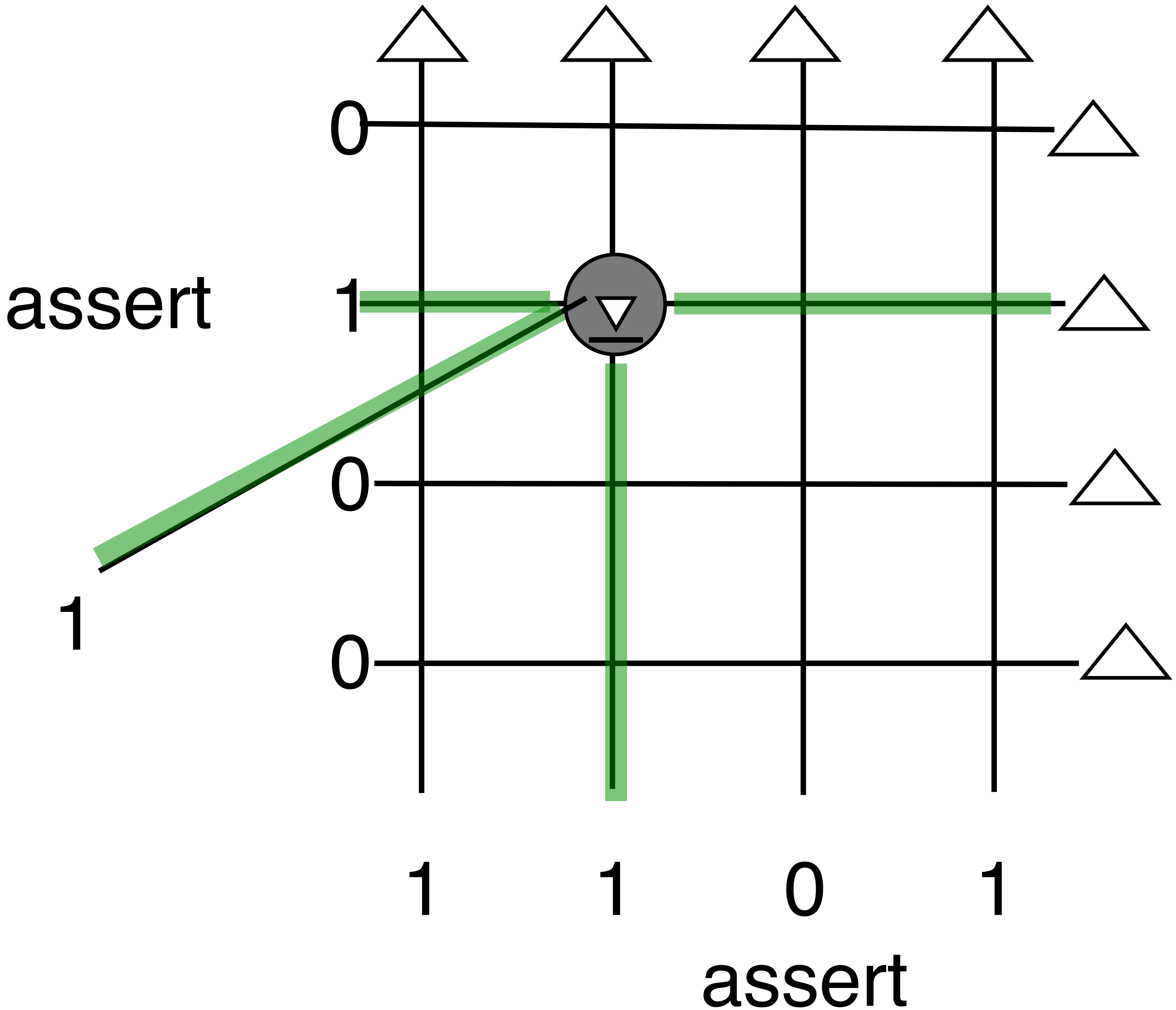
LED Select



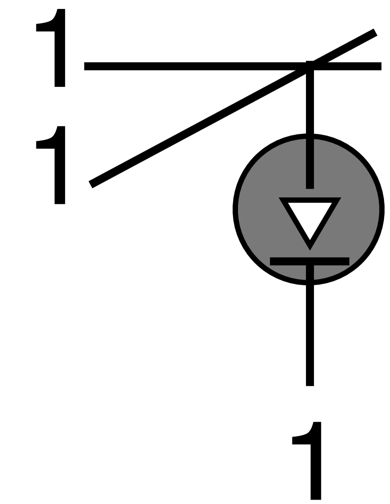
Loop the rows & columns to turn on one LED at a time



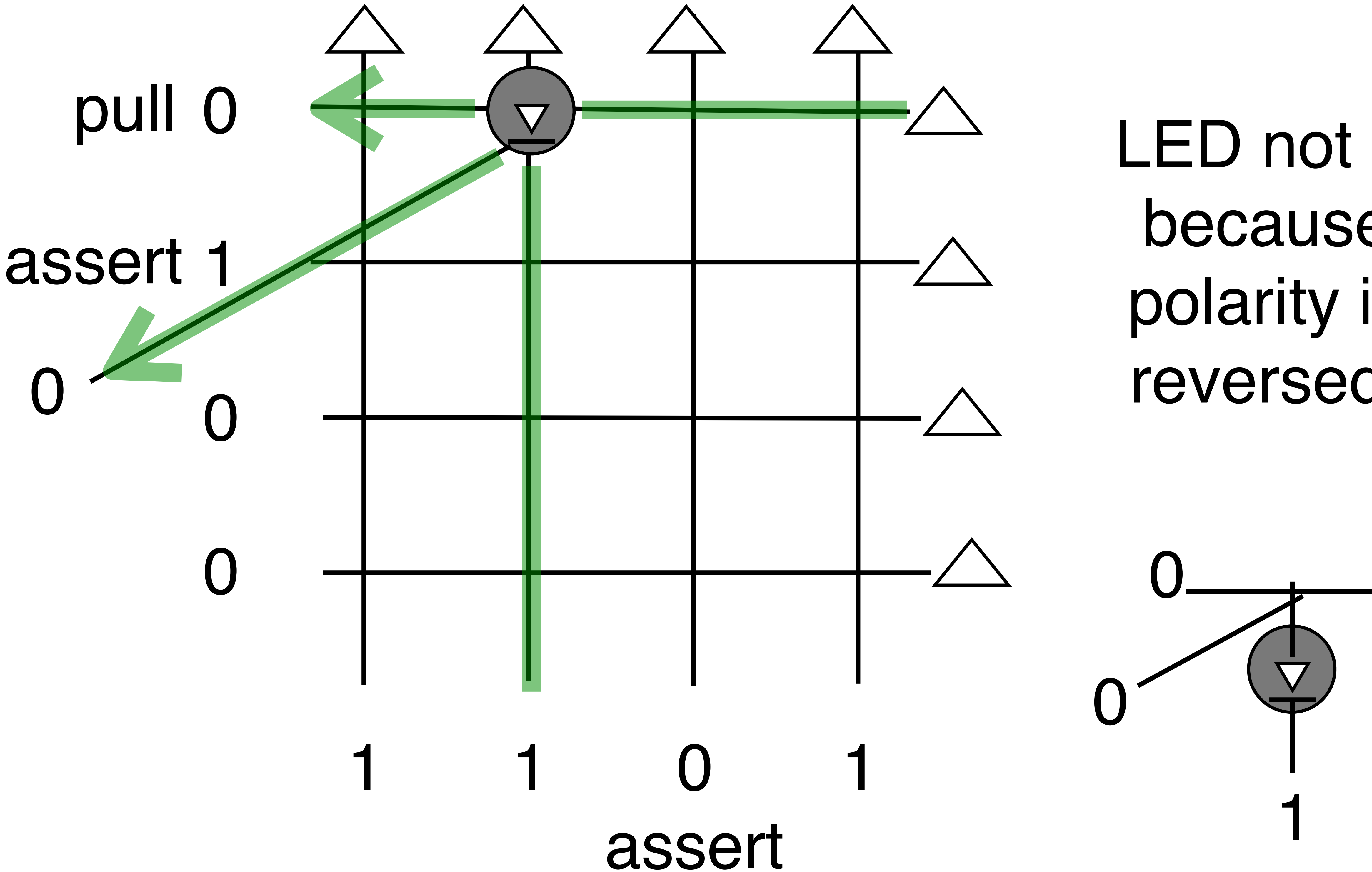
LED unselected: case 1



LED not lit because both sides are of the same high voltage!

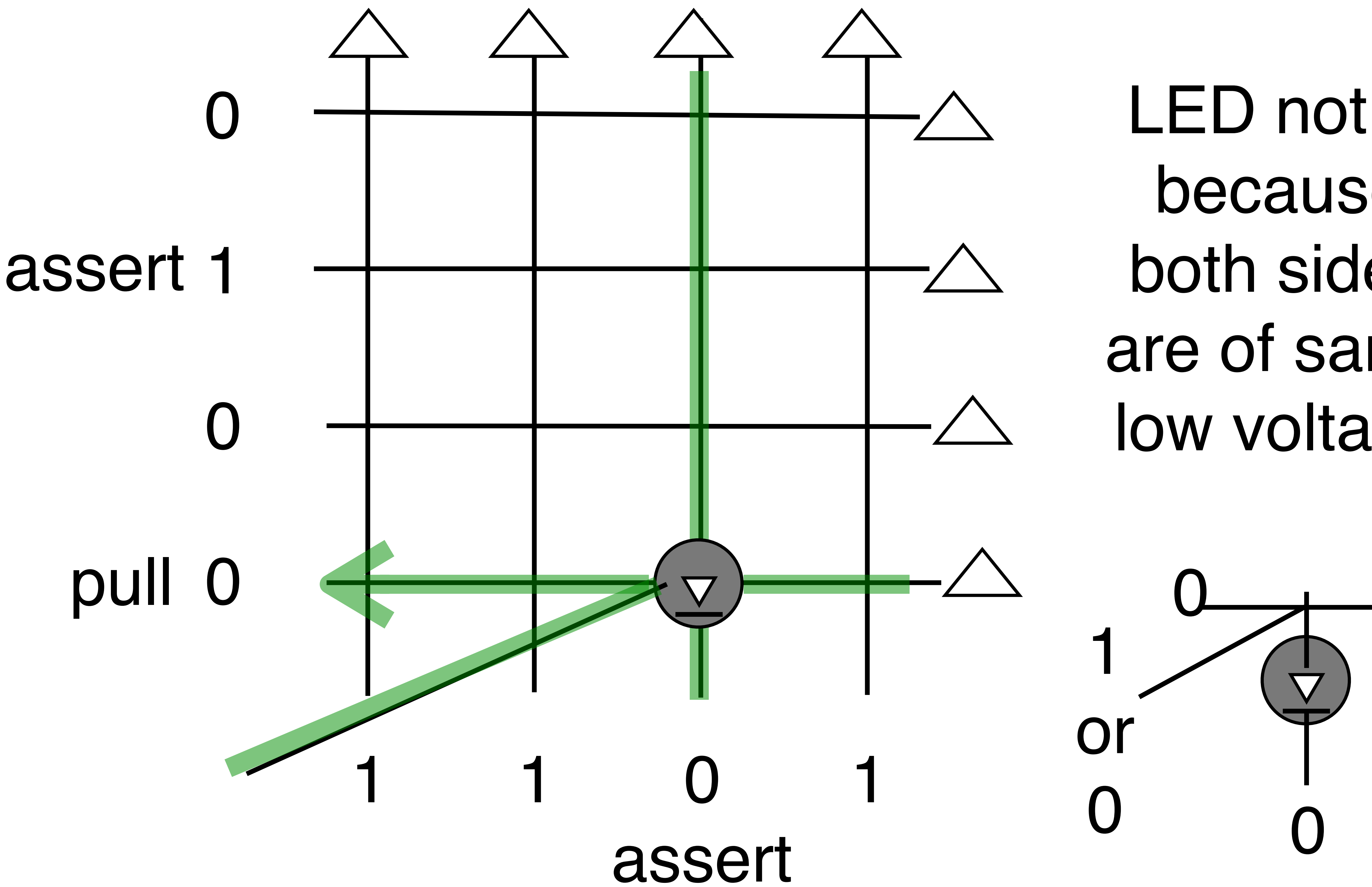


LED unselected: case 2



LED not lit because polarity is reversed!

LED unselected: case 3



LED not lit because both sides are of same low voltage