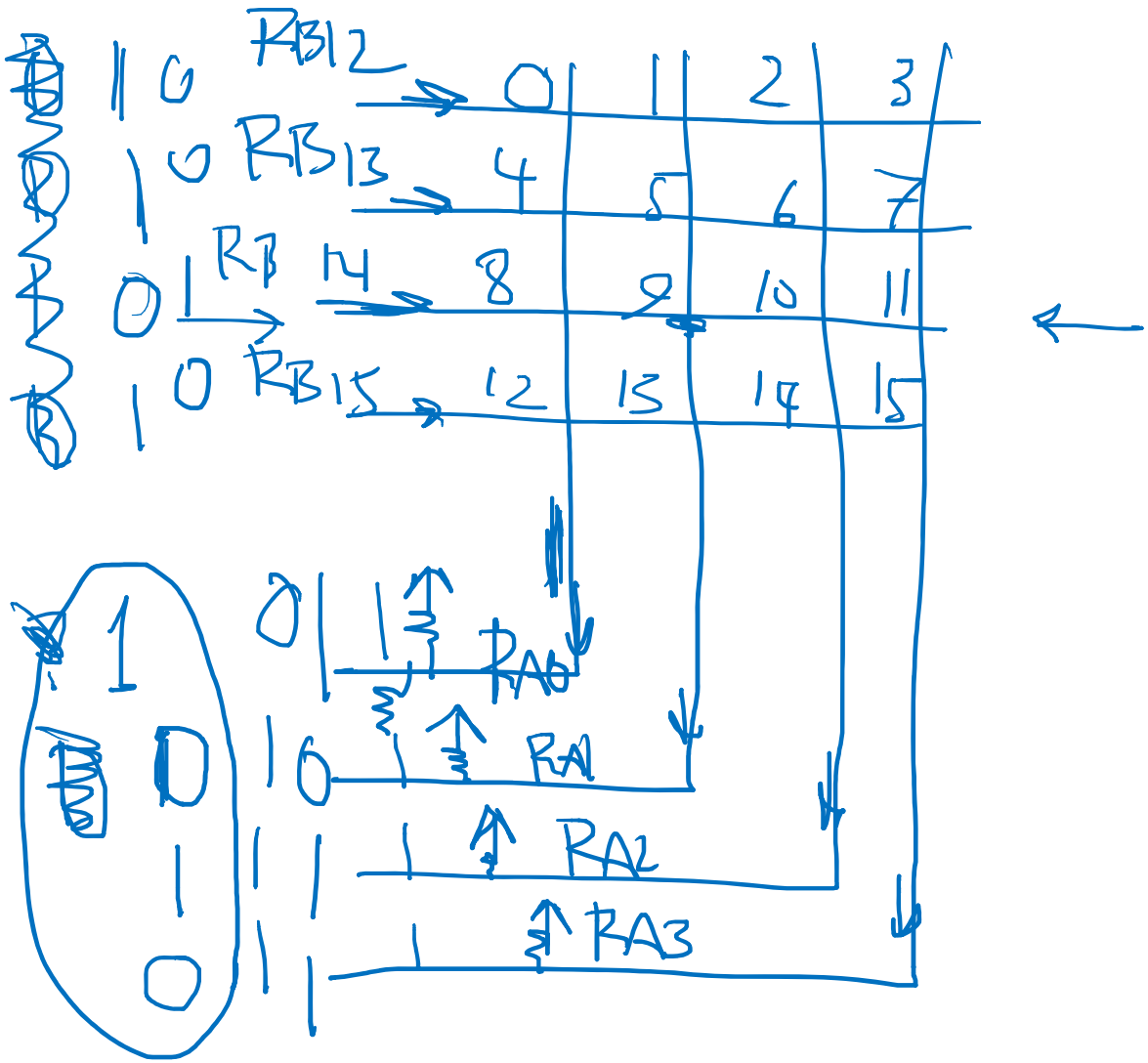
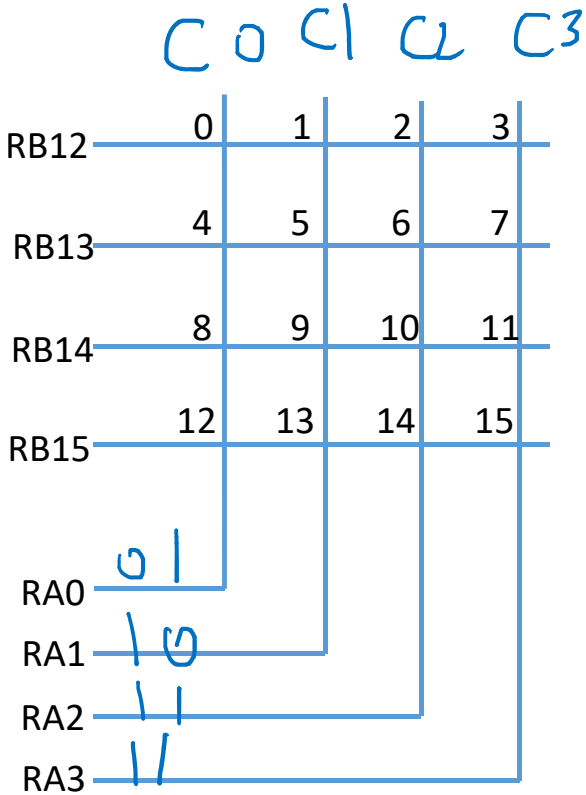

RB12
RB13
RB14
RB15
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
RA0
RA1
RA2
RA3




```
aIn = PORTA;
aIn &= 0xf;
if (aIn == 0b1110)
    Col = 0;
else if (aIn == 0b1101)
    Col = 1;
else
```

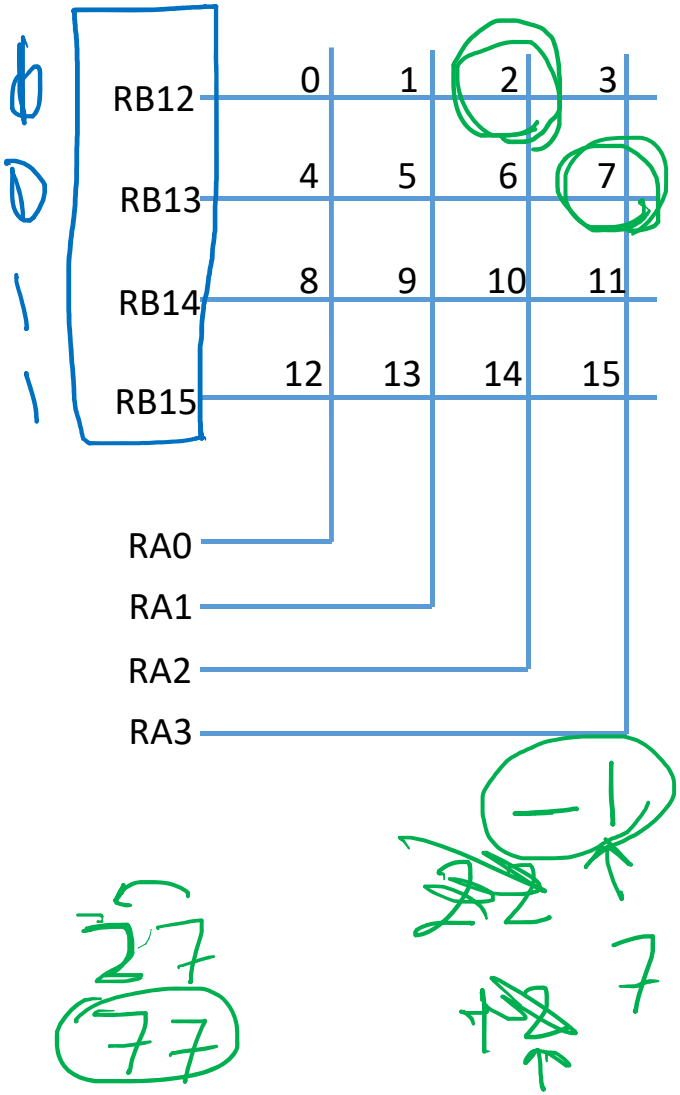
1011

Col = 2

0111

Col = 3

else Col = -1;



LATB = ---- XXXX ---- X

int rowAct[4] = {0b1110 0000 0000 0000,
0b1101 0000 0000 0000,
}

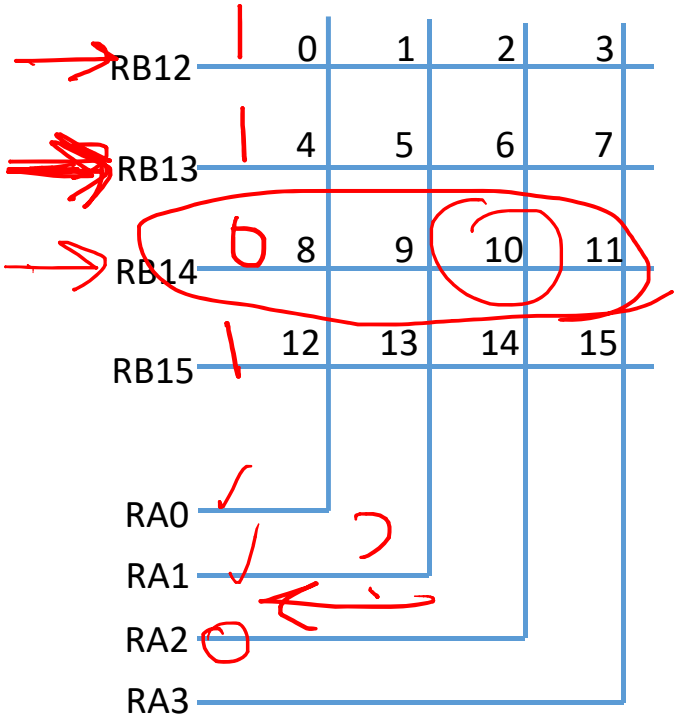
LATB &= ~~0xf000~~ 0x0fff;

LATB |= rowAct[row];

~~row[4] = {0b1110 1111 1111 1111,~~
~~1101 1111 1111 1111~~

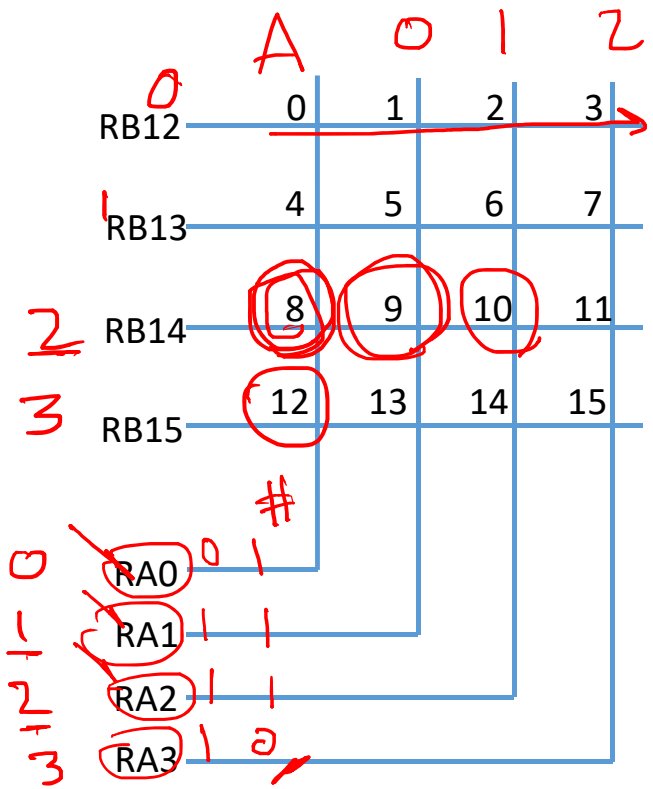
~~LATB |= 0xf000;~~

~~LATB &= rowAct[row]~~




```
for(row=0; row<4; row++){
    LATB &= ~~row~~ 0x0fff;
    LATB |= rowAct[row];

    if(col != -1)
        break;
}
```




```
if (Col==-1)
    KeyIdx=-1;
else
    KeyIdx = row*4+Col;
```

~~Sym = SymTable[KeyIdx];~~

assume row activated

```
#define NOKEY 255

unsigned char scan(unsigned short int row)
{
    static unsigned char LUT[] = {NOKEY,NOKEY,NOKEY,NOKEY,NOKEY,NOKEY,NOKEY,3,
        NOKEY,NOKEY,NOKEY,2,NOKEY,1,0,NOKEY};
    unsigned char key;

    key = PORTA & 15;
    key = LUT[key];
    if (key == NOKEY) return key;
    key += 4*row;
    return  key;
}
```

```
// KBMASK1 is used to mask off the bits in PORTB we don't use (RB12-RB0)
#define KBMASK1 0x0fff
unsigned int keyRowEnPattern[4] = {0xe000, 0xd000, 0xb000, 0x7000};

unsigned char lookupTable[16] = {1, 2, 3, 0xA, 4, 5, 6, 0xB, 7,8,9, 0xC,
    0xe, 0, 0xf, 0xD }; // these numbers need to be further translated
                // to the actual a,b,...,g,dp pattern of the 7-seg LED
                // 0xe translates to '*' and 0xf translates to '#'



.
.
.
   LATB = LATB & KBMASK1;
   LATB |= keyRowEnPattern[row];
```

```c
// not to be used directly in their lab, but students can get ideas...
void waitOneMillisecondAndCheckKeypad(void)
{
    static unsigned short int row = 0;
    static unsigned char lastKey[] = {NOKEY,NOKEY,NOKEY,NOKEY};
    static unsigned short int debounce  = 0;
    unsigned char key;

    // If you haven't covered timers yet, the following two lines create a
    // delay of 1ms
    while (!IFS0bits.T1IF) ;
    IFS0bits.T1IF = 0;

    if (debounce)
    {
        debounce--;
        return;
    }
    row++;
    row &= 3;
    LATB = LATB & KBMASK1;
    LATB |= keyRowEnPattern[row];
```

1ms

40ms

```
    IFS0bits.T1IF = 0;

    if (debounce)
    {
        debounce--;
        return;
    }
    row++;
    row &= 3;
    LATB = LATB & KBMASK1;
    LATB |= keyRowEnPattern[row];

    key = scan(row);
    if (key != lastKey[row])
    {
        lastKey[row] = key;
        if (key != NOKEY)
        {
            // for nonstandard encoding: put( lookupTable[key]);
            put(key);
        }
        debounce = 40; // to give us 40ms debouncing delay (more than enough)
    }
}
```

~~23~~

39

99