Helper macro

```
#define BV(n) 1 <<(n)
```

check nth bit of register

```
if(regr & BV(n))
    // nth bit is 1
else
    // nth bit is 0

regr    :   0x4A    :   0100 1010
        &   BV(3)   :   0000 1000
        ----------------------------
                        0000 1000

regr    :   0x4A    :   0100 1010
        &   BV(4)   :   0001 0000
        ----------------------------
                        0000 0000
```

set nth bit of register

```
regr = regr | BV(n);
or
regr |= BV(n)

regr    :   0x4A    :   0100 1010
        |   BV(4)   :   0001 0000
        -------------------------
                        0101 1010

regr    :   0x4A    :   0100 1010
        |   BV(6)   :   0100 0000
        -------------------------
                        0100 1010
```

clear nth bit of register

```
regr = regr & ~BV(n);
or
regr &= ~BV(n)

regr    :   0x4A    :   0100 1010
```

```
            BV(3)    :   0000 1000
        ~            :   1111 0111
        &
        ---------------------------
                         0100 0010

regr    :   0x4A     :   0100 1010
            BV(5)    :   0010 0000
        ~            :   1101 1111
        &
        ---------------------------
                         0100 1010
```

toggle nth bit of register

```
regr = regr ^ BV(n)
or
regr ^= BV(n)

regr    :   0x4A     :   0100 1010
        ^   BV(3)    :   0000 1000
        -------------------------
                         0100 0010

regr    :   0x4A     :   0100 1010
        ^   BV(4)    :   0001 0000
        -------------------------
                         0101 1010
```

set bits 12 to 15 of register

```
regr = regr | BV(12) | BV(13) | BV(14) | BV(15)
or
regr |=  BV(12) | BV(13) | BV(14) | BV(15)

            BV(12)       :   0000 0000 0000 0000 0001 0000 0000 0000
            BV(13)       :   0000 0000 0000 0000 0010 0000 0000 0000
            BV(14)       :   0000 0000 0000 0000 0100 0000 0000 0000
            BV(15)       :   0000 0000 0000 0000 1000 0000 0000 0000
            |                ---------------------------------------

            |                0000 0000 0000 0000 1111 0000 0000 0000
regr    :   0x00004A00   :   0000 0000 0000 0000 0100 1010 0000 0000
                             ---------------------------------------
                             0000 0000 0000 0000 1111 1010 0000 0000
```

clear bits 17 to 20 of register

```
regr = regr & ~(BV(17) | BV(18) | BV(19) | BV(20))
or
regr &= ~(BV(17) | BV(18) | BV(19) | BV(20))

            BV(17)      :   0000 0000 0000 0010 0000 0000 0000 0000
            BV(18)      :   0000 0000 0000 0100 0000 0000 0000 0000
            BV(19)      :   0000 0000 0000 1000 0000 0000 0000 0000
            BV(20)      :   0000 0000 0001 0000 0000 0000 0000 0000
            |               ---------------------------------------

                            0000 0000 0001 1110 0000 0000 0000 0000
                    ~       1111 1111 1110 0001 1111 1111 1111 1111
regr    :   0x0004A000  :   0000 0000 0000 0100 1010 0000 0000 0000
                            ---------------------------------------
                            0000 0000 0000 0000 1010 0000 0000 0000
```

read value from bit 19 to 24 of register

```
value = (regr >> 19) & 0x0000003F

regr    :   0x104A0000  :   0001 0000 0100 1010 0000 0000 0000 0000
            >> 19       :   0000 0000 0000 0000 0000 0010 0000 1001
        &   0x0000003F  :   0000 0000 0000 0000 0000 0000 0011 1111
        ---------------------------------------------------------
                            0000 0000 0000 0000 0000 0000 0000 1001
```

write value on bit 8 to 15 of register

```
regr &= ~(BV(8) | BV(9) | BV(10) | BV(11) | BV(12) | BV(13) | BV(13) | BV(15));
regr |= (value << 8);

value   :   0x52        :   0000 0000 0000 0000 0000 0000 0101 0010
            << 8        :   0000 0000 0000 0000 0101 0010 0000 0000
regr    :   0x1004A000  :   0001 0000 0000 0100 1010 0000 0000 0000
                        |   0001 0000 0000 0100 0000 0000 0000 0000
                            ---------------------------------------
                            0001 0000 0000 0100 0101 0010 0000 0000
```

wait while bit 4 of register is 0

```
while(regr & BV(4) == 0)
    ;
```

wait while bit 4 of register is 1

```
    while(regr & BV(4) != 0)
        ;
```

Homework

- swap mth and nth bit
- count number of 1's int 32 bit value
- check even or odd parity