Practical bit games

BASIC ASSEMBLY

Assembly language programming
By xorpd

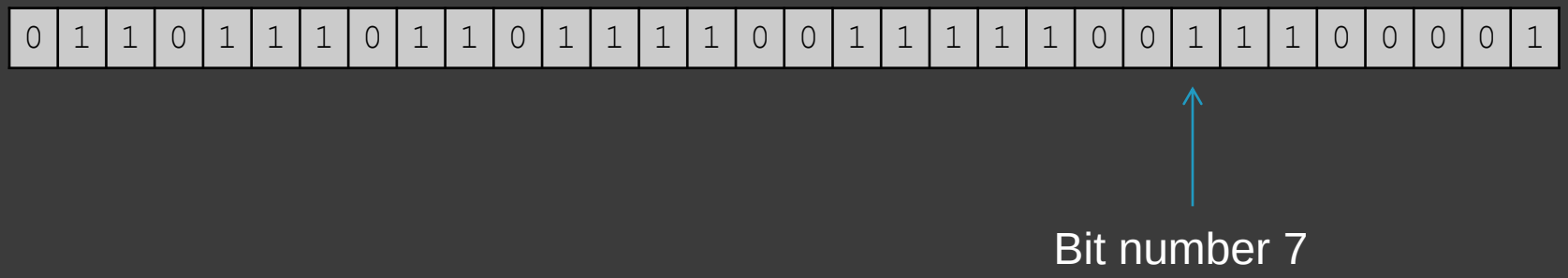
xorpd.net

Objectives

- We will learn about some basic bit manipulation techniques:
  - Extracting one specific bit from a number.
  - Counting the number of “1”-s in a binary number.
  - Calculating modulo powers of two using bit operations.
  - Squeezing many small numbers into the same container.

Extracting bit value

- Challenge:
  - We have a number x, and we want to obtain bit number k. (Leftmost is k=0).



Extracting bit value (Sol 1)

- We will AND with a special “mask”:
  - 00000000 00000000 00000000 10000000

```
        and     eax,00000080h
        jz      bit_7_is_zero
        ; We are here if bit 7 is one.
        jmp     end_if
bit_7_is_zero:
        ; We are here if bit 7 is zero.
end_if:
```

  - Every bit will become zero, except for bit 7 which will be left unchanged.

Extracting bit value (Sol 2)

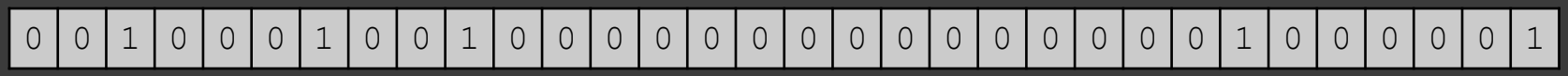
- The carry flag is a copy of the last bit that was “kicked out”.

```
        ror     eax,8
        ; bit 7 is copied into the Carry flag.
        jnc     bit_7_is_zero
        ; We are here if bit 7 is one.
        jmp     end_if
bit_7_is_zero:
        ; We are here if bit 7 is zero.
end_if:
        rol     eax,8    ; restore eax
```

- We can later use ROL to restore the original value of eax.

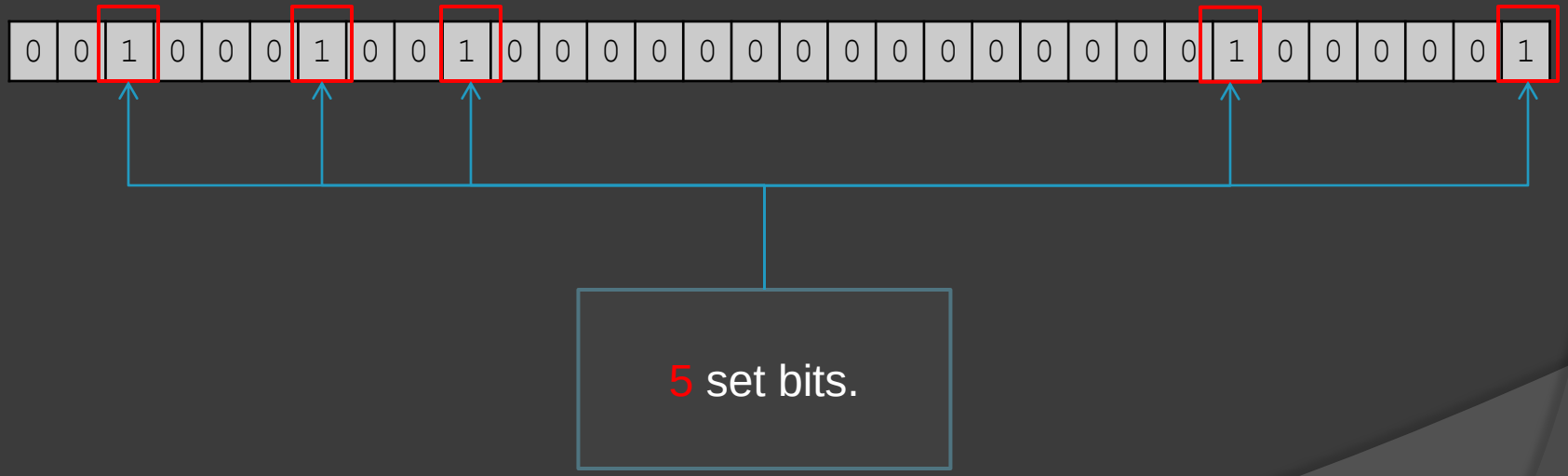
Counting set bits

- Question:
  - How many set bits (1 bits) are there in some number x? (“Population count”)



Counting set bits

- Question:
  - How many set bits (1 bits) are there in some number x? (“Population count”)



Counting set bits (Sol 1)

- We extract every bit from eax using AND, and sum all the bits.

```
        mov     edx,0    ; edx counts "1" bits.
        mov     ecx,32d  ; 32 bits.
sum_bits:
        mov     ebx,eax
        and     ebx,1    ; Take lowest bit.
        add     edx,ebx  ; Count lowest bit.
        ror     eax,1    ; Rotate to next bit.
        loop    sum_bits
```

Counting set bits (Sol 2)

- We rotate eax 32 times, each time checking the carry flag.

```
        mov     edx,0    ; edx counts "1" bits.
        mov     ecx,32d  ; 32 bits.
count_bits:
        ror     eax,1    ; lsb is copied to the CF.
        jnc     bit_is_zero
        inc     edx      ; Increase count of "1"-s.
bit_is_zero:
        loop    count_bits
```

Calculating modulo

- We want to calculate $x \,\%\, 2^k$.
  - We could use DIV
    - but it is a slow instruction.
  - We can take advantage of the bit structure of the number.
    - $x = b_0 \cdot 2^0 + b_1 \cdot 2^1 + b_2 \cdot 2^2 + b_3 \cdot 2^3 + \cdots$
    - $2^k \mid b_m \cdot 2^m$ for $m \geq k$.
    - $b_0 \cdot 2^0 + b_1 \cdot 2^1 + \cdots + b_{k-1} 2^{k-1} < 2^k$.
    - It is enough to consider the lowest k bits.

Calculating modulo (Sol)

- Calculating modulo 4 (last 2 bits):

```
and      eax,11b
; 11b = 3
```

- Calculating modulo 64 (last 6 bits):

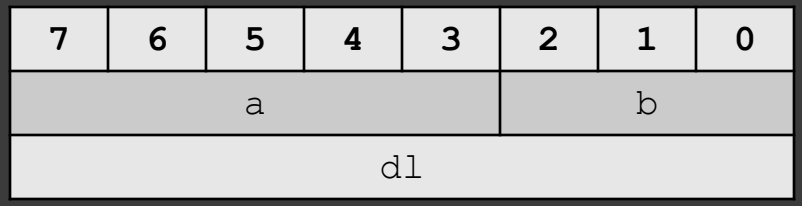
```
and      eax,111111b
; 111111b = 63
```

- This method only works for calculating modulo of powers of two.

Data Packing

- Challenge:
  - We have two numbers inside al, bl.
  - $al < 2^5, bl < 2^3$.
  - We want to squeeze those two numbers into dl.
    - It should be possible, dl is of size 8 bits.

Data Packing (Sol)



- Packing:

```
; al < 2^5, bl < 2^3
mov     dl,al
shl     dl,3     ; Make room for b.
or      dl,bl    ; xor will work too.
```

- Unpacking:

```
; dl contains two packed numbers.
mov     cl,dl    ; Make a copy of dl
and     dl,111b  ; Take lowest 3 bits (b)
mov     bl,dl
shr     cl,3     ; Take highest 5 bits (a)
mov     al,cl
```

Summary

- We have learned about:
  - Extracting one specific bit from a number.
  - How to count the amount of set bits in a given number.
  - How to calculate modulo 2^k without using DIV.
  - How to pack two small numbers into one register, and how to unpack them.

Exercises

- Code reading.
  - New subroutine **print_eax_binary**.
- Code writing.