SPRAWOZDANIE

ARCHITEKTURA KOMPUTERÓW 2 - projekt

Czytnik kodów kreskowych

WYKONAWCA: *Bartłomiej Sawicki, Szymon Hutnik*

WYDZIAŁ: *W4, Elektryczny*

TERMIN ODDANIA ETAPU 2: *10.05.2021, 00.00*

DATA ODDANIA: *09.05.2021, 16.00*

PROWADZĄCY: *dr Dominik Żelazny*

GITHUB: *https://github.com/ketrab100/ak-projekt*

1. Opis projektu

Napisanie programu w assemblerze czytającego kod kreskowy zapisany w plikach graficznych o różnych formatach np. bmp, jpg, gif.

2. Postępy

a) Zaimplementowane:

- otwieranie oraz odczyt plików bmp,
- obliczanie ilości pixeli oraz offsetu i paddingu,
- sprawdzanie czy pixel jest czarny w RGB 24bit.

b) Do zaimplementowania:

- dokończenie funkcji zliczającej pixele
- odczytywanie cyfry po zliczonych pixelach
- rozszerzenie programu o kolejne zakresy kolorów
- rozszerzenie programu o kolejne formaty graficzne

3. Kluczowe elementy programu

a) Obsługa plików

```
# open file
    movl $5, %eax                                   # sys_open
    movl $name, %ebx                                #name to ebx
    movl $0, %ecx                                   # access: read-only
    movl $0777, %edx                                # read, write and execute by all
    int $0x80                                       # system call
```

```
# read from file
    movl $3, %eax                                  # sys_read
    movl fd, %ebx                                  # filedecryptor to ebx
    movl $buf, %ecx                                # buf to ecx (ecx = .string "0")
    movl $len, %edx                                # lenght of file to edx
    int $0x80                                      # system call
```

b) Obliczanie bajtów paddingu

```
# calcPaddingBytes
    movl $3, %eax                           # 3 to eax (3 bytes per color)
    mull sizex                              # eax *= sizex
    movl %eax, %ecx                         # eax to ecx
    clc                                     # clear carry flag
    movl $4, %ebx                           # 4 to ebx
    divl %ebx                               # eax/=4
    mull %ebx                               # eax*=4
    subl %ecx, %eax                         # eax -= ecx  ([3*sizex/4*4] - sizex*3 == 3*sizex%4-4)
    cmp $0, %eax                            # if eax==0
    je padding0                             # go to padding 0
    addl $4, %eax                           # eax+=4
    movl %eax, paddingBytes                 # paddingBytes=eax
    jmp endCalcPaddingBytes                 # go to endCalcPaddingBytes
padding0:
    movl $0, paddingBytes                   # 0 to paddingBytes
endCalcPaddingBytes:
```

c) Liczba bajtów w jednym rzędzie z uwzględnieniem paddingu

```
# get pixelsPerBar
    movl DIBHeaderSize, %eax                    # DIBHeaderSize to eax
    addl $14, %eax                              # add 14 to eax (header is always 14)
    movl %eax, offset                           # beginning of pixels

    movl $3, %eax                               # 3 to eax (3 bytes per color)
    mull sizex                                  # eax = number of bytes per row
    movl %eax, %ecx                             # eax to ecx
    movl $2, %ebx                               # 2 to ebx
    movl sizey, %eax                            # sizey to eax
    divl %ebx                                   # eax over ebx - middle row (number of pixels/2), saved in eax
    mull %ecx                                   # eax times ecx - number of starting byte of middle row

    addl %eax, offset                           # offset+=eax - total offset

    movl $2, %ebx                               # 2 to ebx
    movl sizey, %eax                            # sizey to eax
    divl %ebx                                   # eax over ebx - middle row (number of pixels/2), saved in eax
    mull paddingBytes                           # eax*=paddingBytes

    addl %eax, offset                           # offset+=eax
    movl offset, %edi                           # offset to edi
```

4. Napotkane problemy

Napisanie funkcji obliczającej paddingu sprawiło nam największy problem, gdyż operacja modulo w assemblerze jest dużo bardziej złożona niż *„a%b”*. Sporą trudnością jest wyszukiwanie informacji w internecie gdyż większość stron o assemblerze operuje w składni intela.