

PRŮBĚŽNÁ PREZENTACE PŘEDMĚTU OS

**JAKUB VÁVERKA
DAVID BOHMANN
VÁCLAV JANOCH**

ZADÁNÍ

Simulace operačního systému DOS

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

Welcome to DOSBox v0.74

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount
Current mounted drives are:
Drive Z is mounted as Internal Virtual Drive

Z:\>mount c C:\Users\Jakub
Drive C is mounted as local directory C:\Users\Jakub\

Z:\>C:
C:\>_
```



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

1DOLLAR-TX

C:\DOCUMENTS\OS\COM>_
```

CO TO VLASTNĚ DĚLÁ?

```
40 decrypt:
41
42     mov al, byte ptr ds:[edx]
43     mov al, byte ptr es:[esi+eax]
44     add byte ptr ds:[bx], al
45
46     inc ebx
47     inc edx
48
49     dec cx
50     cmp cx, 0
51     jnz decrypt
```

3144

Convert Reset Swap

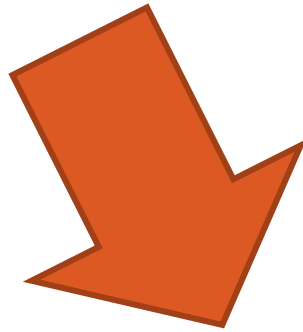
1D

Select

```
77 key1 db 20h, 3ah, 23h, 32h, 0bh, 3dh, 1fh, 13h, 4ch, 19h, 05h, 07h, 07h, 00h
78 key2 db 00h, 0ah, 11h, 08h, 03h, 1dh, 1ah, 08h, 04h, 2ch, 3fh, 33h, 1ah, 41h
79 key3 db 01h, 00h, 08h, 0bh, 0ch, 07h, 0ah, 05h, 02h, 09h, 06h, 03h, 04h, 00h
```

CO MÁME?

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
1DOLLAR-TX
C:\DOCUME~1\OS\COM>_
```



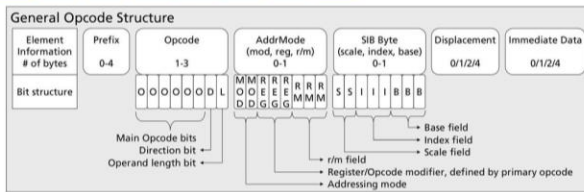
```
C:\Windows\System32\cmd.exe
1DOLLAR-TX
C:\Users\Jakub\.CLion2016.2\system\cmake\generated\DosEmulator-c406b4fd\c406b4fd\Debug1>
```

JAK TO ŠLO?

x86 Opcode Structure and Instruction Overview

1st	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADD					ES PUSH SS	ES POP SS	OR					CS PUSH DS	TWO BYTE POP DS		
1	ADC							SBB					POP DS			
2	AND					ES SEGMENT OVERRIDE	DAA	SUB					CS SEGMENT OVERRIDE	DAS		
3	XOR					SS	AAA	CMP					DS	AAS		
4	INC							DEC								
5	PUSH							POP								
6	PUSHAD	POPAD	BOUND	ARPL	FS	GS	OPERAND SIZE SEGMENT OVERRIDE	ADDRESS SIZE SIZE OVERRIDE	PUSH	IMUL	PUSH	IMUL	INS	OUTS		
7	JO	JNO	JB	JNB	JE	JNE	JBE	JA	JS	JNS	JPE	JPO	JL	JGE	JLE	JG
8	ADD/ADC/AND/XOR OR/SBB/SUB/CMP			TEST		XCHG		MOV REG			MOV SREG	LEA	MOV SREG	POP		
9	NOP	XCHG EAX					CWD			CDQ	CALL/FWAIT	PUSHFD	POPFD	SAHF	LAHF	
A	MOV EAX			MOVSI		CMPS		TEST		STOS		LODS		SCAS		
B	MOV															
C	SHIFT IMM		RETN		LES	LDS	MOV IMM		ENTER	LEAVE	RETF		INT3	INT IMM	INTO	IRETD
D	SHIFT 1		SHIFT CL		AAM		AAD	SALC	XLAT		FPU					
E	LOOPNZ	LOOPZ	LOOP	JECXZ		IN IMM		OUT IMM		CALL	JMP	JMPF	JMP SHORT	IN DX		OUT DX
F	LOCK EXCLUSIVE ACCESS	ICE BP	REPNE REPE		HLT		CMC	TEST/NOT/NEG [I]MUL/[I]DIV		CLC	STC	CLI	STI	CLD	STD	INC/DEC CALL/IMP PUSH

	Arithmetic & Logic		Prefix
	Memory		System & I/O
	Stack		No Operation (NOP) / Multiple Instructions / Extended Instruction Set
	Control Flow & Conditional		



	2nd		1st																					
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F								
0	(L,S)LODT (L,S)STR (L,S)VER(L,R,W)		(L,S)GDT (L,S)IDT (L,S)MSW		LAR		LSL				CLTS		INVD		WBINVD		UD2				NOP			
1	SSE{1,2,3}								Prefetch SSE1				HINT_NOP											
2	MOV CR/DR								SSE{1,2}															
3	WRMSR		RDTS		RDMSR		RDPMS		SYSENTER		SYSEXIT		GETSEC SMX		MOVBE / THREE BYTE		THREE BYTE SSE4							
4	CMOV																							
5	SSE{1,2}																							
6	MMX, SSE2																							
7	MMX, SSE{1,2,3}, VMX													MMX, SSE{2,3}										
8	JO	JNO	JB	JNB	JE	JNE	JBE	Jcc SHORT		JA	JS	JNS	JPE	JPO	JL	JGE	JLE	JG						
9	SETO	SETNO	SETB	SETNB	SETE	SETNE	SETBE	SETA	SETS	SETNS	SETPE	SETPO	SETL	SETGE	SETLE	SETG								
	SETcc																							
A	PUSH FS	POP FS	CPUID	BT	SHLD				PUSH GS	POP GS	RSM	BTS	SHRD		*FENCE	IMUL								
B	CMPXCHG	LSS	BTR	LFS	LGS	MOVZX		POPCNT	UD	BT BTS BTR BTC		BTC	BSF	BSR	MOVSB									
C	XADD		SSE{1,2}					CMPXCHG		BSWAP														
D	MMX, SSE{1,2,3}																							
E	MMX, SSE{1,2}																							
F	MMX, SSE{1,2,3}																							

Addressing Modes									
mod	0		01		10		11		
r/m	32bit	32bit	16bit	32bit	16bit	32bit		32bit	r/m / REG
000	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]			AL / AX / EAX
001	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]			CL / CX / ECX
010	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]			DX / EDI / EDI
011	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]			BL / BX / EBX
100	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]			SI / SP / ESI
101	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]			DI / BP / EBP
110	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]			SI / DI / ESI
111	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]	[RAX]			BI / DI / EBI

encoding	scale (2bit)	Index (3bit)	Base (3bit)
000	2 ⁰ =1	[EAX]	EAX
001	2 ¹ =2	[ECX]	ECX
010	2 ² =4	[EDX]	EDX
011	2 ³ =8	[EBX]	EBX
100	--	none	ESP
101	--	[EBP]	(disp32 / double / QWORD) / disp16 / WORD
110	--	[ESI]	ESI
111	--	[EDI]	EDI

SIB value = index * scale + base

v1.0 – 30.08.2011

Contact: Daniel Plohmann – +49 228 73 54 228 – daniel.plohmann@fkf.fraunhofer.de

Source: Intel x86 Instruction Set Reference

Opcode table presentation inspired by work of Ange Albertini

CO BYLO EASY?

Skok do sebe

```
32 ;nasledujuci kod nastavi eax na 1  
33 ; je to jmp -1 nasledovany inc eax, který je ted 0  
34 db 0ebh, 0ffh, 0c0h
```

CO REKTILO?

Nexorovaný segmentový register

```
16  mov dx,cs
17  mov ds,dx
18  mov es, dx
19
20  ;zacneme s desifrovanim
21
22  xor eax, eax
23  xor ecx, ecx
24  xor ebx, ebx
25  xor edx, edx
26  xor esi, esi
```

Přímý zápis do grafického bufferu

```
63  mov word ptr [di+146], 794Bh ;K
64  mov word ptr [di+148], 7949h ;I
65  mov word ptr [di+150], 7956h ;V
66  mov word ptr [di+152], 792Fh ;/
67  mov word ptr [di+154], 794Fh ;O
68  mov word ptr [di+156], 7953h ;S
69  mov word ptr [di+158], 7921h ;!
```

**DĚKUJEME VÁM ZA
POZORNOST**

Nějaké otázky?