ОНЛАЙН-ОБРАЗОВАНИЕ

OTUS

Регистры процессора, работа с памятью

Особенности работы ПК

Пакулов Артур

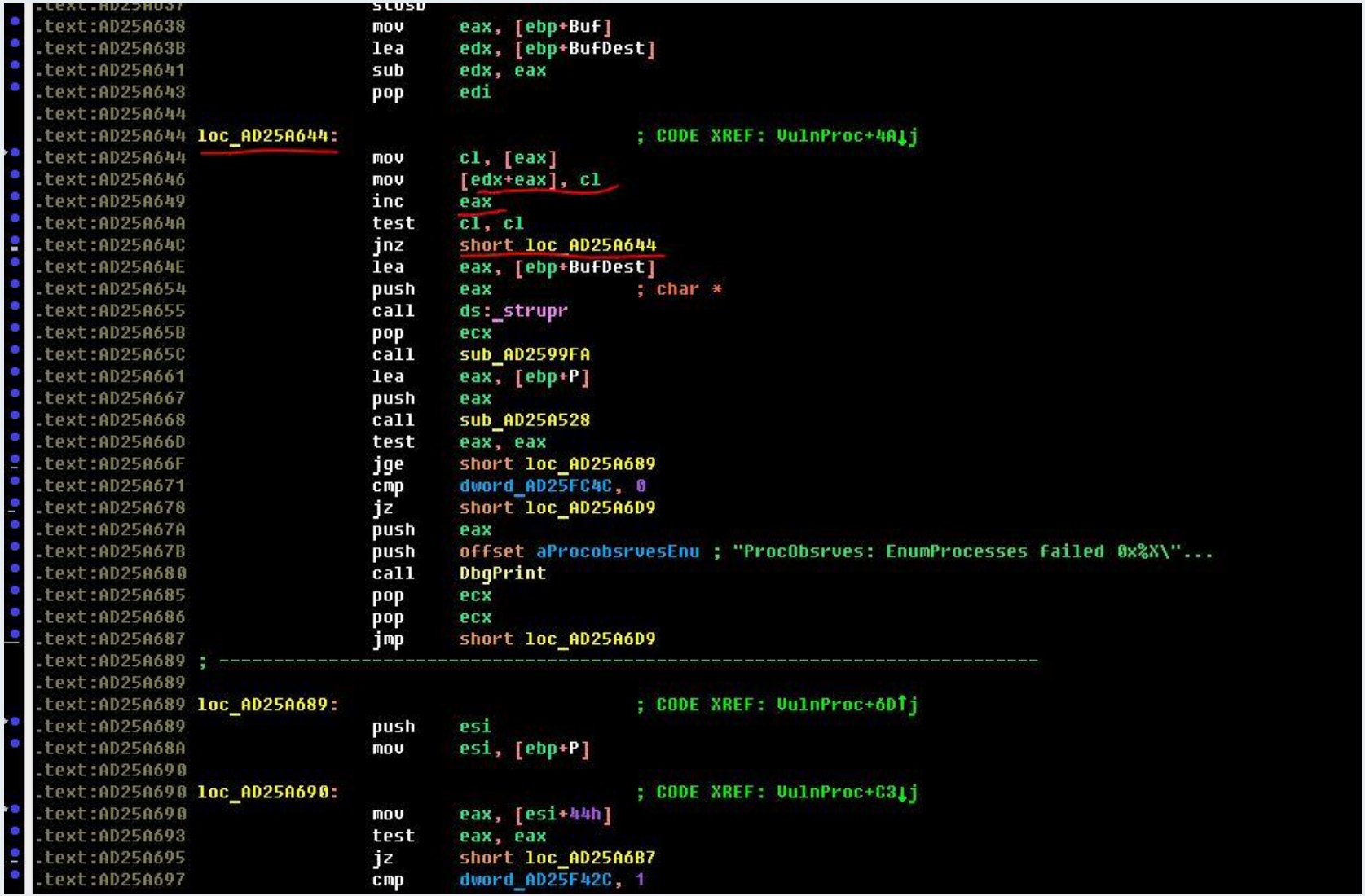
А зачем оно надо?

Мотивация

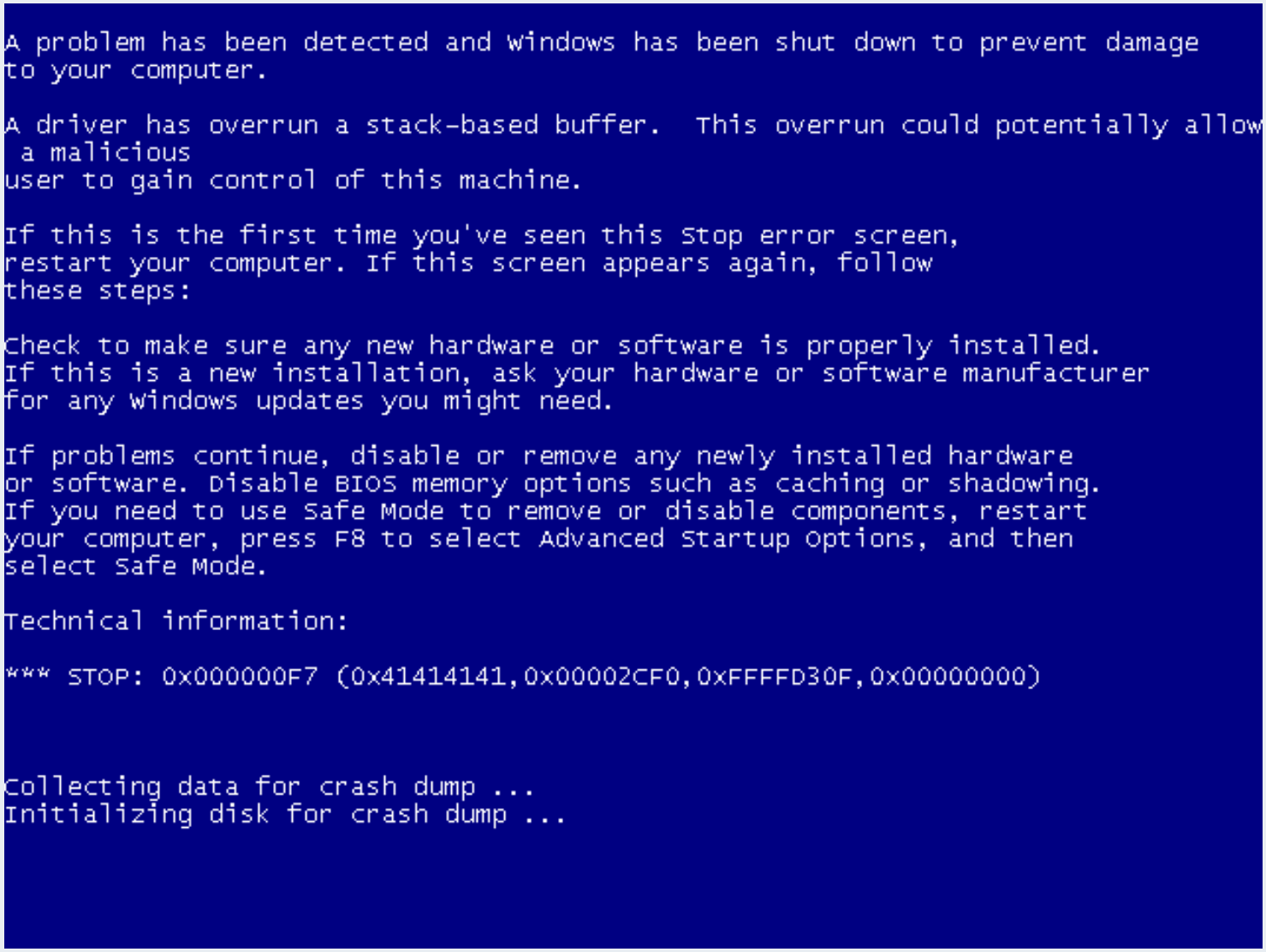
А зачем оно надо?

```
                mov     [esi+4], ebx

label_getMem:                           ; CODE XREF: proc5_1+A6↑j
                mov     ax, [esi]       ; если eax = 0xfffe,то
                                        ; add eax,2 даст 0!
                add     ax, 2
                mov     [esi+2], ax
                jz      short loc_8BCC4A1B
                movzx   eax, ax
                push    edi             ; Tag
                add     eax, 8
                push    eax             ; NumberOfBytes
                push    ebx             ; PoolType
                call    ds:ExAllocatePoolWithTag
                mov     [esi+4], eax
                cmp     eax, ebx
                jz      loc_8BCC4946
                movzx   ecx, word ptr [esi+2]
                mov     [eax], ecx
                mov     eax, [esi+4]
                mov     [eax+4], edi
                movzx   ecx, word ptr [esi+2]
                add     dword ptr [esi+4], 8
                mov     eax, [esi+4]
                push    ecx             ; size_t
                push    ebx             ; int
                push    eax             ; void *
                call    memset
                add     esp, 0Ch
                jmp     short label_copy
; ---------------------------------------------------------------------------

loc_8BCC4A1B:                           ; CODE XREF: proc5_1+F3↑j
                mov     [esi+4], ebx

label_copy:                             ; CODE XREF: proc5_1+96↑j
                                        ; proc5_1+131↑j
                movzx   eax, word ptr [esi]
                push    eax             ; size_t
                push    [ebp+inBuff]    ; void *
                push    dword ptr [esi+4] ; void *
                call    memcpy
                movzx   eax, word ptr [esi]
```

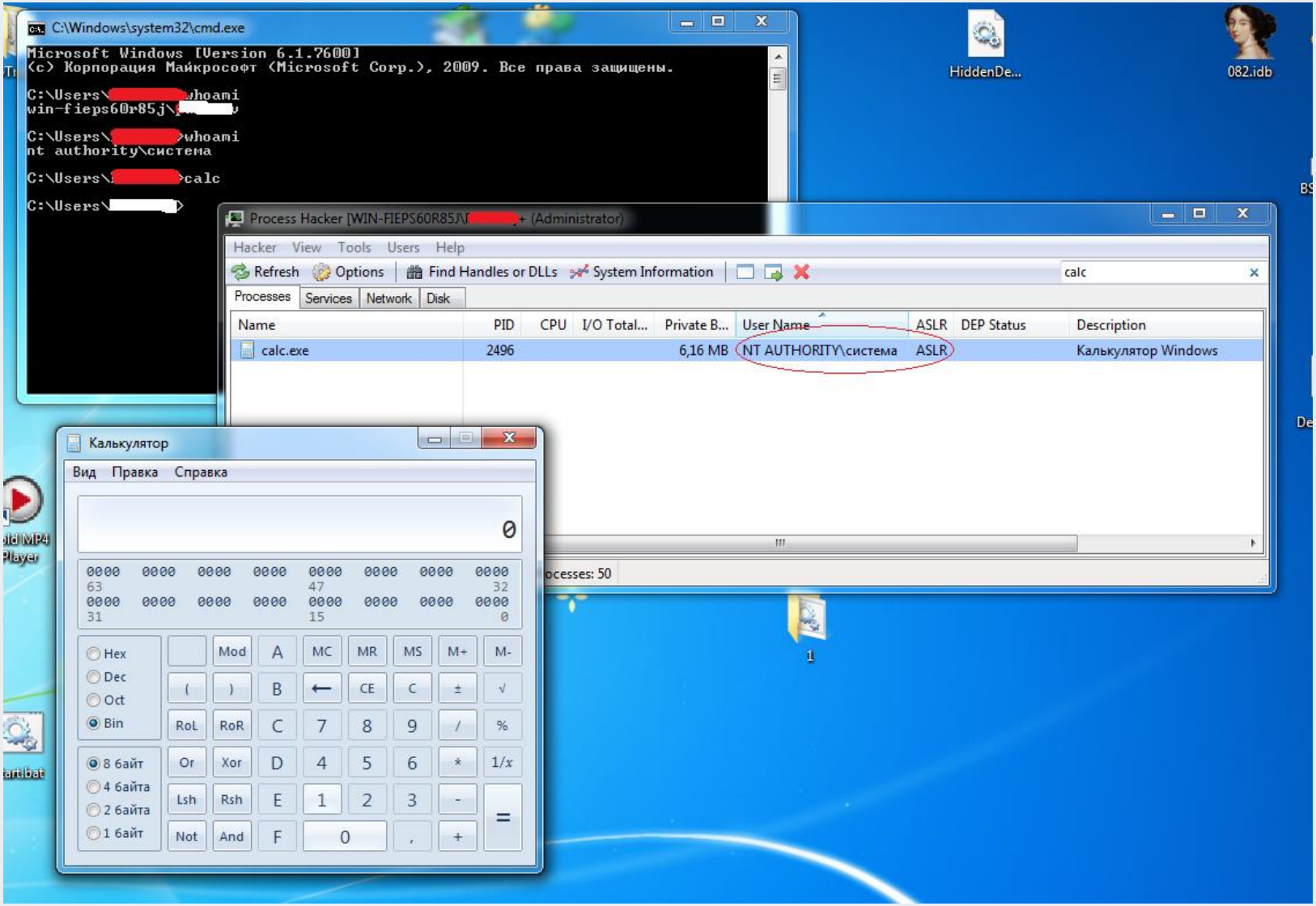
А зачем оно надо?



А зачем оно надо?



А зачем оно надо?



А зачем оно надо?

Линейка процессоров 8086

OTUS

- 1971 год. Работал с 4х (!) разрядными данными
- 8080 – 1974 год. 8 разрядные данные, 64к память
- 8086 – 1978 год. 16 разрядные данные, 1М память
- 8088 – 1979 год. 16 разрядные данные, 1М память
- 80186 – 1983 год.
- 80286 – 1983 год. 16 разрядные данные, 16М память
- 80386 – 1983 год. 32 разрядные данные, 4Г память
- I386 – 1987 год.
- I486 – 1990 год.
- Pentium – 1993 год. 64 разрядные данные.

Особенности работы ПК

Оперативная память

Особенности работы ПК

Регистры общего назначения

AX (accumulator)	AH/AL
CX (count)	CH/CL
DX (data)	DH/DL
BX (base)	BH/BL
Si (Source index)	
Di (Destination Index)	
Bp (Base Pointer)	
Sp (Stack pointer)	

Сегментные регистры

- CS (Code Segment)
- DS (Data Segment)
- SS (Stack Segment)
- ES (Extra Segment)

Instruction Pointer

IP

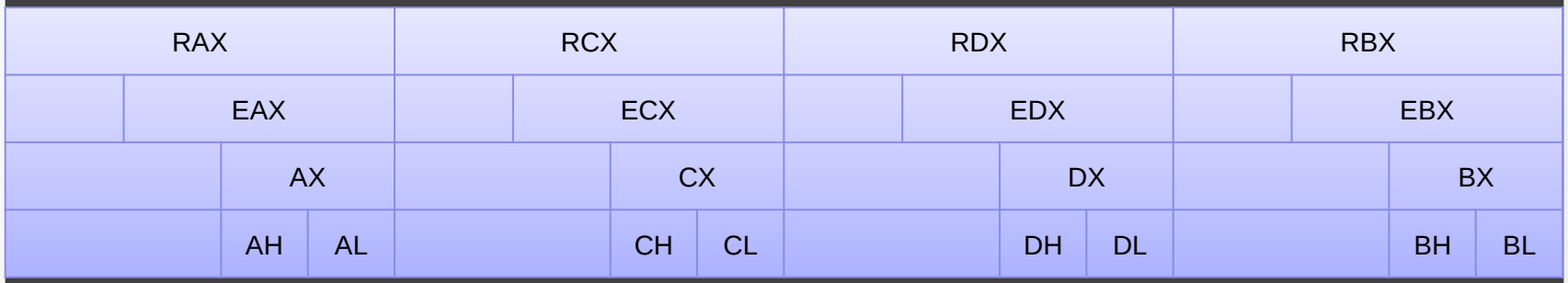
Регистр флагов

Флаги условий: cf, of, zf, sf, pf, af

Флаги состояний: df, if, tf

64-х разрядные процессоры

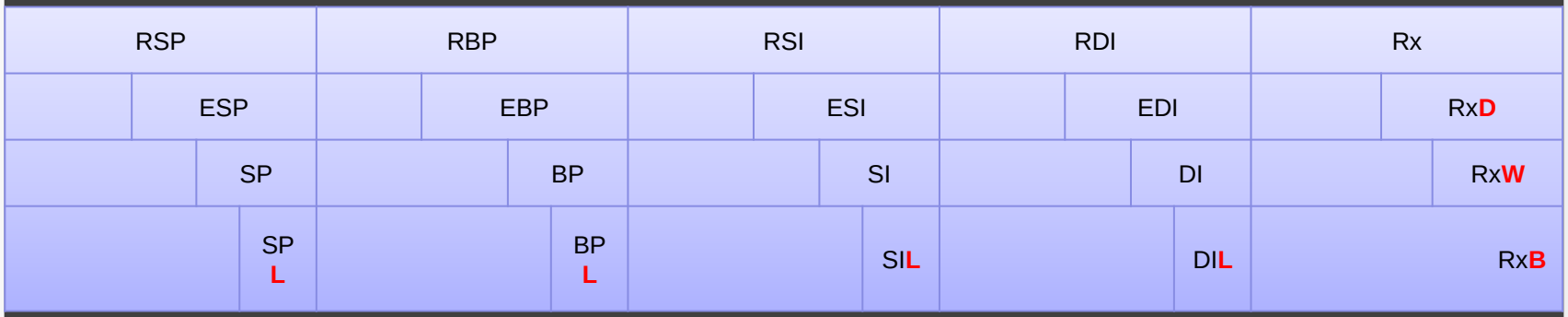
16 регистров общего назначения



64-х разрядные процессоры

RDI, RSI, RBP и RSP расширены для побайтного доступа!

К R1-R15 можно обращаться как к **D**WORD, **W**ORD, **B**YTE



X – цифра [0-15]

Зачем пишут x64 малварь?

Если архитектура Windows поддерживает выполнение x32 кода на x64 OS системе

Режимы работы процессора

OTUS

- Реальный режим (Real Mode)
- Режим системного управления (System Management Mode)
- Виртуальный режим i8086 (V86)
- Защищенный режим (Protected Mode)

Модель памяти

В 8086 процессоре

Плоская модель памяти

OTUS

Что интересного?

- Сегментация
- Логический адрес
- Физический адрес

Плоская модель памяти

ВЫВОДЫ:

Номер сегмента всегда кратен 0x10 ➔ заканчивается на 0

Последняя цифра не записывается в сегментный регистр

Сегментные регистры всегда 16и разрядный

Физический адрес = segment * 0x10 + offset

Плоская модель памяти

Диапазон адресов: 0 - 2^20 -1 = [0; 0xFFFFF]

```
mov si, 0xFFFF

mov bx, [0xFFFF+0xF]

mov cx, [si+0xF]
```

```
00000000: BEFFFF                mov         si,-1 ;'  '
00000003: 678B1D0E000100        mov         bx,[00001000E]
0000000A: 8B4C0F                mov         cx,[si][00F]
```

Плоская модель памяти

Сегментация сокращает размер команд!

```
00000000: BEFFFF                  mov         si,-1 ;'  '
00000003: 678B1D0E000100          mov         bx,[00001000E]
0000000A: 8B4C0F                  mov         cx,[si][00F]
```

Плоская модель памяти

OTUS

А что, если выражение получится > 0xFFFFF ?

Физический адрес = segment * 0x10 + offset

Адресная линия A20

Адресная шина процессора 8086 – 20и разрядная. Может адресовать 1Mb памяти

Диапазон логических адресов процессора 8086 – [0000h:0000h – 0xffff:0xffff]

НО!

Тогда, диапазон Физических адресов:
[0 – 0xffff*0x10+0xffff] = [0 – 0x10ffef

20 разрядов с [0-19]

Адресная линия A20

Преобразование логического адреса в физический происходит по модулю 0xfffff

Тогда, диапазон логических адресов:
[0000h:0000h – 0xffff:0x000f]

Физический адрес максимального логического:
[0 – 0xffff*0x10+0x000f] = [0 - 0xfffff]

Адресная линия A20

Все физические адреса, полученные из выходящих из диапазона логических «оборачиваются» (& 0xFFFFF)

Пример:

[0000h:0000h – 0xffff:0x10]

✓ [0 – 0xffff*0x10+0x10] = [0 – 0x100000 & 0xFFFFF] = [0 – 0x00000]

Адресная линия A20

Итог:

Jmp 0:0 ~ jmp 0xFFFF:0x10

В реальном режиме 8086 используются младшие 20 адресных линий: A0-A19

Т.К Более новые процессоры имеют ширину шины > 20, в реальном режиме адреса не оборачиваются, что приводит к нарушению обратной совместимости, поэтому адресная линия A20 по умолчанию блокируется

ДЗ

Tool chains

OTUS

Emu 8086

FASM

X64 Dbg

Ida Pro

HIEW

FAR MANAGER & CONEmu

OTUS

Вопросы???

Пакулов Артур

A.Pakulov.Otus@Gmail.com

Спасибо
за внимание!