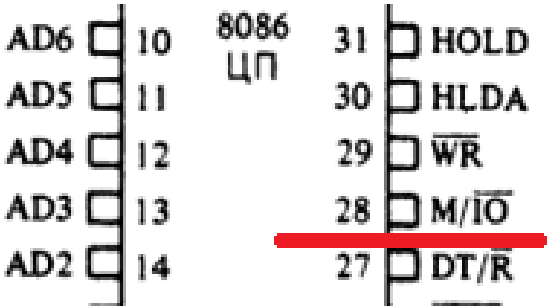
Машинно-зависимые языки программирования

Савельев Игорь Леонидович



POST card

OUT port, reg

OUT DX, reg

Номер порта – [0, 255] или [0, 65535]

Размер данных (reg) – **8**, 16, 32 бита

План

- Цепочечные примитивы
  - MOVS*, LODS*, SCAS*, CMPS*
  - REP / REPE(Z) / REPNE(Z)

Адресация в памяти

[**reg + reg*scale + const**]

reg – база, базовый адрес
req* - индекс
const – смещение
scale – 1, 2, 4, 8

Общий вид исполнительного адреса

Адресация в памяти

```
[750]                   ; displacement only
[rbp]                   ; base register only
[rcx + rsi*4]           ; base + index * scale
[rbp + rdx]             ; scale is 1
[rbx - 8]               ; displacement is -8
[rax + rdi*8 + 500]     ; all four components
[rbx + counter]         ; uses the address of the variable
                        ;'counter' as the displacement
```

```
void *memcpy(void *dest, const void *src, size_t n);
```

Функция memcpy() копирует n байтов из области памяти src в область памяти dest. Области памяти не могут пересекаться. Используйте memmove(3), если области памяти перекрываются.

```
void *memcpy(void *dest, const void *src, size_t n)
{
    for (size_t i = 0; i < n; ++i )
    {
        *(dest + i) = *(src + i);
        // dest[i] = src[i];
    }
    return dest;
}
```

1. Цепочечные команды

```
section .data
    src  db 44h,33h,22h,11h
    dest dd 0

section .text
0   xor eax, eax
    mov ecx, 0
    mov edi, dest
    mov esi, src
next:
1   mov al, [esi]     ; al = *(src + i)
    mov [edi], al     ; *(dest+i) = al
2   inc ecx           ; i++
    inc esi           ; src + i
    inc edi           ; dest + i
3   cmp ecx, 4        ; sizeof(dd) == 4
    jne next
    ret
```

	0	1	2	3
al	0	0x44	0x44	0x44
esi	0x404040	0x404040	0x404041	0x404041
edi	0x404044	0x404044	0x404045	0x404045
ecx	0	0	1	1
dest[]	0,0,0,0	44h,0,0,0	44h,0,0,0	44h,0,0,0
zf	1	1	1	0
al		0x33	0x33	0x33
esi		0x404041	0x404042	0x404042
edi		0x404045	0x404046	0x404046
ecx		1	2	2
dest[]		44h,33h,0,0	44h,33h,0,0	44h,33h,0,0
zf		0	0	0
al		0x22	0x22	0x22
esi		0x404042	0x404043	0x404043
edi		0x404046	0x404047	0x404047
ecx		2	3	3
dest[]		44h,33h,22h,0	44h,33h,22h,0	44h,33h,22h,0
zf		0	0	0
al		0x11	0x11	0x11
esi		0x404043	0x404044	0x404044
edi		0x404047	0x404048	0x404048
ecx		3	4	4
dest[]		44h,33h,22h,11h	44h,33h,22h,11h	44h,33h,22h,11h
zf		0	0	1

1. Цепочечные команды

Синтаксис:	**MOVSB**
Операнды:	Нет
Назначение:	Копирование строк байтов
Процессор:	8086+
Флаги:	Не изменяются
Комментарий:	Команда MOVSB копирует один байт из ячейки памяти по адресу DS:(E)SI в ячейку памяти по адресу ES:(E)DI . Для 64 битного режима из (R|E)SI в (R|E)DI. После выполнения команды, регистры SI и DI увеличиваются на 1, если флаг DF = 0, или уменьшаются на 1, если DF = 1.

1. Цепочечные команды

```
section .data
    src  dd 0x11223344
    dest dd 0

section .text

    xor eax, eax
    mov ecx, 0
    mov edi, dest
0   mov esi, src
    cld         ; направление
next:
1   movsb     ; *(dest+i) = *(src + i)
2   inc ecx     ; i++
3   cmp ecx, 4 ; sizeof(dd) == 4
    jne next   ; jle next
```

	0	1	2	3
esi	0x404040	0x404041	0x404041	0x404041
edi	0x404044	0x404045	0x404045	0x404045
ecx	0	0	1	1
dest[]	0,0,0,0	44h,0,0,0	44h,0,0,0	44h,0,0,0
zf	1	1	1	0
esi		0x404042	0x404042	0x404042
edi		0x404046	0x404046	0x404046
ecx		1	2	2
dest[]		44h,33h,0,0	44h,33h,0,0	44h,33h,0,0
zf		0	0	0
esi		0x404043	0x404043	0x404043
edi		0x404047	0x404047	0x404047
ecx		2	3	3
dest[]		44h,33h,22h,0	44h,33h,22h,0	44h,33h,22h,0
zf		0	0	0
esi		0x404044	0x404044	0x404044
edi		0x404048	0x404048	0x404048
ecx		3	4	4
dest[]		44h,33h,22h,11h	44h,33h,22h,11h	44h,33h,22h,11h
zf		0	0	1

1. Цепочечные команды

```
void *memcpy(void *dest, const void *src, size_t n)
{
    for (size_t i = 0; i < n; ++i )
    {
        *(dest + i) = *(src + i);
        // dest[i] = src[i];
    }
    return dest;
}
```

Изменим направление

```
void *memcpy(void *dest, const void *src, size_t n)
{
    for (size_t i = n - 1; i >= 0; --i )
    {
        *(dest + i) = *(src + i);
        // dest[i] = src[i];
    }
    return dest;
}
```

1. Цепочечные команды

```
section .data
    src  dd 0x11223344
    dest dd 0

section .text

    xor eax, eax
    mov ecx, 3
    mov edi, dest
    add edi, ecx
    mov esi, src
0   add esi, ecx
    std
next:
1   movsb       ; *(dest+i) = *(src + i)
2   dec ecx     ; i--
3   cmp ecx, 0 ;
    jge next
```

	0	1	2	3
esi	0x404043	0x404042	0x404042	0x404042
edi	0x404047	0x404046	0x404046	0x404046
ecx	3	3	2	2
dest[]	0,0,0,0	0,0,0,11h	0,0,0,11h	0,0,0,11h
sf==of	t	t	t	t
esi		0x404041	0x404041	0x404041
edi		0x404045	0x404045	0x404045
ecx		2	1	1
dest[]		0,0,22h,11h	0,0,22h,11h	0,0,22h,11h
sf==of		t	t	t
esi		0x404040	0x404040	0x404040
edi		0x404044	0x404044	0x404044
ecx		1	0	0
dest[]		0,33h,22h,11h	0,33h,22h,11h	0,33h,22h,11h
sf==of		T	T	t
esi		0x40403F	0x40403F	0x40403F
edi		0x404043	0x404043	0x404043
ecx		0	0xFFFFFFFF	0xFFFFFFFF
dest[]		44h,33h,22h,11h	44h,33h,22h,11h	44h,33h,22h,11h
sf==of		t	t	f

Размер данных кратен 2

```
void *memcpy(void *dest, const void *src, size_t n)
{
    for (size_t i = 0; i < n/2; ++i )
    {
        *(((uint16_t*)dest) + i) = *(((uint16_t*)src) + i);
    }
    return dest;
}
```

1. Цепочечные команды

Синтаксис:	**MOVSW**
Операнды:	Нет
Назначение:	Копирование строк слов
Процессор:	8086+
Флаги:	Не изменяются
Комментарий:	Команда MOVSW копирует слово (2 байта) из ячейки памяти по адресу DS:(E)SI в ячейку памяти по адресу ES:(E)DI . Для 64 битного режима из (R|E)SI в (R|E)DI. После выполнения команды, регистры SI и DI увеличиваются на 2, если флаг DF = 0, или уменьшаются на 2, если DF = 1.

1. Цепочечные команды

```
section .data
    src  dd 0x11223344
    dest dd 0

section .text

    xor eax, eax
    mov ecx, 0
    mov edi, dest
0   mov esi, src
    cld                 ; направление
next:
1   movsw               ;
2   inc ecx             ; i ++ !!!
3   cmp ecx, 2          ;  = 4/2
    jne next
```

	0	1	2	3
esi	0x404040	0x404042	0x404042	0x404042
edi	0x404044	0x404046	0x404046	0x404046
ecx	0	0	1	1
dest[]	0,0,0,0	44h,33h,0,0	44h,33h,0,0	44h,33h,0,0
zf	1	1	1	0
esi		0x404044	0x404044	0x404044
edi		0x404048	0x404048	0x404048
ecx		1	2	2
dest[]		44h,33h,22h,11h	44h,33h,22h,11h	44h,33h,22h,11h
zf		0	0	1

Размер данных кратен 4

```
void *memcpy(void *dest, const void *src, size_t n)
{
    for (size_t i = 0; i < n/4; ++i )
    {
        *(((uint32_t*)dest) + i) = *(((uint32_t*)src) + i);
    }
    return dest;
}
```

1. Цепочечные команды

Синтаксис:	**MOVSD**
Операнды:	Нет
Назначение:	Копирование строк двойных слов
Процессор:	80386+
Флаги:	Не изменяются
Комментарий:	Команда MOVSD копирует двойное слово (4 байта) из ячейки памяти по адресу DS:(E)SI в ячейку памяти по адресу ES:(E)DI . Для 64 битного режима из (R|E)SI в (R|E)DI. После выполнения команды, регистры SI и DI увеличиваются на 4, если флаг DF = 0, или уменьшаются на 4, если DF = 1.

1. Цепочечные команды

```
section .data
    src  dq 0x112233445667788
    dest dq 0

section .text

    xor eax, eax
    mov ecx, 0
    mov edi, dest
0   mov esi, src
    cld      ; направление
next:
1   movsd    ;
    inc ecx  ; i ++ !!!
2   cmp ecx, 2 ;  = 8/4
    jne next
    ret
```

	0
esi	0x404040
edi	0x404048
ecx	0
dest[]	0,0,0,0,0,0,0,0
zf	1

	1	2
esi	0x404044	0x404044
edi	0x404048	0x404048
ecx	0	1
dest[]	88h,77h,66h,55h,0,0,0,0	88h,77h,66h,55h, 0,0,0,0
zf	1	0
esi	0x404048	0x404048
edi	0x40404A	0x40404A
ecx	1	2
dest[]	88h,77h,66h,55h,44h,33h,22h,11h	88h,77h,66h,55h,44h,33h,22h,11h
zf	0	1

Размер данных кратен 8

```
void *memcpy(void *dest, const void *src, size_t n)
{
    for (size_t i = 0; i < n/8; ++i )
    {
        *(((uint64_t*)dest) + i) = *(((uint64_t*)src) + i);
    }
    return dest;
}
```

1. Цепочечные команды

Синтаксис:	**MOVSQ**
Операнды:	Нет
Назначение:	Копирование строк двойных слов
Процессор:	64 бита
Флаги:	Не изменяются
Комментарий:	Команда MOVSQ копирует четыре слова (8 байт) из ячейки памяти по (R|E)SI в ячейку памяти по адресу (R|E)DI. После выполнения команды, регистры SI и DI увеличиваются на 8, если флаг DF = 0, или уменьшаются на 4, если DF = 1.

1. Цепочечные команды

```
section .data
    src  dq 0x112233445667788
         dq 0x112233445667788
    dest dq 0,0

section .text

    xor eax, eax
    mov ecx, 0
    mov edi, dest
0   mov esi, src
    cld
next:
1   movsq    ;
    inc ecx   ; i ++ !!!
2   cmp ecx, 2 ;   = 16/4
    jne next
    ret
```

	0
esi	0x404040
edi	0x404050
ecx	0
dest[]	0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0
zf	0

	1	2
esi	0x404048	0x404048
edi	0x404058	0x404058
ecx	0	1
dest[]	88h,77h,66h,55h,44h,33h,22h,11h 0,0,0,0,0,0,0,0	88h,77h,66h,55h,44h,33h,22h,11h 0,0,0,0,0,0,0,0
zf	0	0
esi	0x404050	0x404050
edi	0x404060	0x404060
ecx	1	2
dest[]	88h,77h,66h,55h,44h,33h,22h,11h 88h,77h,66h,55h,44h,33h,22h,11h	88h,77h,66h,55h,44h,33h,22h,11h 88h,77h,66h,55h,44h,33h,22h,11h
zf	0	1

1. Цепочечные команды

```
size_t strlen(const char *s);
```

Функция strlen() вычисляет длину строки s. Завершающий символ `\0'не учитывается.

```
size_t strlen(const char *s)
{
    size_t cnt = 0;

    while( *(s + cnt) )
       ++cnt;

    return cnt;
}
```

```
size_t strlen(const char *s)
{
    int cnt = -1;

    do
    {
        cnt ++;
    } while( *(s + cnt) != 0 );

    return (size_t)cnt;
}
```

1. Цепочечные команды

```
size_t strlen(const char *s) {
    size_t cnt = 0;

    while( *(s + cnt) )
       ++cnt;
    return cnt;
}
```

	0
edi	0x404040
eax	0
byte [edi + eax]	h
zf	1

```
section .data
    dest  db "hello", 0

section .text

        mov     edi, dest
0       xor     eax, eax
@b:
1       cmp     byte[edi+eax],0
        je      @f
2       inc     eax             ;результат
        jmp     @b
@f:
```

	1	2	1	2
eax	0	1	1	2
byte [edi + eax]	h	e	e	l
zf	0	0	0	0
eax	2	3	3	4
byte [edi + eax]	l	l	l	o
zf	0	0	0	0
eax	4	5	5	
byte [edi + eax]	o	0	0	
zf	0	0	1	

1. Цепочечные команды

Синтаксис:	**SCASB**
Операнды:	Нет
Назначение:	Сканирование строки байт
Процессор:	8086+
Флаги:	Флаги OF, SF, ZF, AF, PF, CF устанавливаются в соответствии с результатом.
Комментарий:	Команда SCASB сравнивает регистр AL с байтом в ячейке памяти по адресу DS:(E)DI и устанавливает флаги аналогично команде CMP. После выполнения команды, регистр DI увеличивается на 1, если флаг DF = 0, или уменьшается на 1, если DF = 1. Если команда используется в 64-разрядном режиме адресации, то используется регистр (R|E)DI.

1. Цепочечные команды

```
size_t strlen(const char *s){
    int cnt = -1;

    do {
        cnt ++;
    } while( *(s + cnt) != 0 );

    return (size_t)cnt;
}
```

```
section .data
    dest  db "hello", 0
section .text
    xor eax, eax
    mov ecx, 0
    mov edi, dest
0   dec ecx
    cld
next:
1   inc ecx
    mov al, 0 ;
2   scasb       ; *(dest+i) == al ?
    jne next
```

	0
edi	0x404040
al	0
ecx	0xFFFFFFFF
byte [edi]	h
zf	1

	1	2	1	2	1	2
edi	0x404040	0x404041	0x404041	0x404042	0x404042	0x404043
ecx	0	0	1	1	2	2
[edi]	h	h (e)	e	e (l)	l	l (l)
zf	1	0	0	0	0	0
edi	0x404043	0x404044	0x404044	0x404045	0x404045	0x404046
ecx	3	3	4	4	5	5
[edi]	l	l (o)	o	o (0)	0	0 (~)
zf	0	0	0	0	0	1

1. Цепочечные команды

интаксис:	**SCASW**
Операнды:	Нет
Назначение:	Сканирование строки слов
Процессор:	8086+
Флаги:	Флаги OF, SF, ZF, AF, PF, CF устанавливаются в соответствии с результатом.
Комментарий:	Команда SCASW сравнивает регистр AX со словом в ячейке памяти по адресу ES:(E)DI и устанавливает флаги аналогично команде CMP. После выполнения команды, регистр DI увеличивается на 2, если флаг DF = 0, или уменьшается на 2, если DF = 1. Если команда используется в 64-разрядном режиме адресации, то используется регистр (R|E)DI.

1. Цепочечные команды

```
section .data
    dest  dw 0x11, 0x0000
          dw 0x22, 0x0000
section .text
    xor eax, eax
    xor ebx, ebx   ; счетчик
                   ; цикла
    mov ecx, 0     ; счетчик
                   ; нулей
0   mov edi, dest
    cld
step:
    mov ax, 0
1   scasw      ; *(dest+i)==ax?
    jne next
2   inc ecx
next:
3   inc ebx
4   cmp ebx, 4
    jne step
```

	0
edi	0x404040
ebx	0
ecx	0
word[edi]	0x11
zf	1

	1	3	4	1	2	3	4
edi	0x404042	0x404042	0x404042	0x404044	0x404044	0x404044	0x404044
ebx	0	1	1	1	1	2	2
ecx	0	0	0	0	1	1	1
[edi]	0x11 (0x00)	0x00	0x00	0x00 (0x22)	0x00 (0x22)	0x22	0x22
zf	0	0	0	1	0	0	0
edi	0x404046	0x404046	0x404046	0x404048	0x404048	0x404048	0x404048
ebx	2	3	3	3	3	4	4
ecx	1	1	1	1	2	2	2
[edi]	0x22 (0x00)	0x00	0x00	0x00 (~)	~	~	~
zf	0	0	0	1	0	0	1

1. Цепочечные команды

Синтаксис:	**SCASD**
Операнды:	Нет
Назначение:	Сканирование строки двойных слов
Процессор:	8086+
Флаги:	Флаги OF, SF, ZF, AF, PF, CF устанавливаются в соответствии с результатом.
Комментарий:	Команда SCASD сравнивает регистр EAX с двойным словом в ячейке памяти по адресу ES:(E)DI и устанавливает флаги аналогично команде CMP. После выполнения команды, регистр DI увеличивается на 4, если флаг DF = 0, или уменьшается на 4, если DF = 1. Если команда используется в 32-разрядном режиме адресации, то используется регистр (R|E)DI.

:: toUpper (char *src, size_t n);

```
toUpper (char *src,  size_t n)
{
    int i = 0;
    while ( n )
    {
        *(src + i) = *(src + i) - 0x20;
        // *(src + i) = *(src + i) & 0xDF;  1101.1111
        i ++;
        n --;
    }
}
```

1. Цепочечные команды

:: toUpper (char *src, size_t n);

```
section .data
    src  db "hello"
    n    db  $ - src

section .text

    xor eax, eax
    mov ecx, [n]  ; strlen("hello")
    mov edi, src
    mov esi, src
step:
    mov al, [esi]
    inc esi          ;
    and al, 0xDF     ; sub al, 0x20
    mov [edi], al
    inc edi          ;

    dec ecx
    cmp ecx, 0
    jne step
```

h = 0x68, H = 0x48
e = 0x65, E = 0x45
l = 0x6C, L = 0x4C
o = 0x6F, O = 0x4F

1. Цепочечные команды

Синтаксис:	**LODSB**
Операнды:	Нет
Назначение:	Чтение байта из строки
Процессор:	8086+
Флаги:	Не изменяются
Комментарий:	Команда LODSB копирует байт из памяти по адресу DS:(E)SI в регистр AL. После выполнения команды, регистр SI увеличивается на 1, если флаг DF = 0, или уменьшается на 1, если DF = 1. Если команда используется в 64-разрядном режиме адресации, то используется регистр (R|E)SI.

1. Цепочечные команды

:: toUpper (char *src, size_t n);

```
section .data
    src  db "hello"
    n    db  $ - src

section .text

    xor eax, eax
    mov ecx, [n]  ; strlen("hello")
    mov edi, dest
    mov esi, dest
    cld
step:
    lodsb             ; al = *(src + i)
    and al, 0xDF      ; to Uppercase (sub al, 0x20)
    mov [edi], al
    inc edi           ;

    dec ecx
    cmp ecx, 0
    jne step
```

int strcmp (const char *_s1_, const char *_s2_);

```
int strcmp (const char *s1, const char *s2)
{
    int ret = 0;

    while (*(s1 + ret) || *(s2 + ret))
    {
        if(*(s1 + ret) != *(s2 + ret))
        {
           ret ++;
           break;
        }
    }
    return ret;
}
```

1. Цепочечные команды

int strcmp (const char *_s1_, const char *_s2_);

```
        mov     esi, lpStr1     ; Указатели на строки
        mov     edi, lpStr2
        xor     eax,eax          ; Предположим, что строки равны
next:
1       lodsb                    ; Сравнить следующие символы
2       mov     ah,[edi]
        inc     edi

3       or      al,al            ; Первая строка закончилась?
        jz      @first              ; Да
4       cmp     al,ah            ; Символы совпадают?
        je      next               ; Да, проверить следующий символ
@first:
5       or      ah,ah            ; Вторая строка закончилась?
        je      @end               ; Да, строки равны

6       mov     eax, 1          ; Строки не совпадают
@end:
```

1. Цепочечные команды

Синтаксис:	**LODSW**
Операнды:	Нет
Назначение:	Чтение слова из строки
Процессор:	8086+
Флаги:	Не изменяются
Комментарий:	Команда LODSW копирует слово из памяти по адресу DS:(E)SI в регистр AX. После выполнения команды, регистр SI увеличивается на 2, если флаг DF = 0, или уменьшается на 2, если DF = 1. Если команда используется в 64-разрядном режиме адресации, то используется регистр (R|E)SI.

1. Цепочечные команды

Синтаксис:	**LODSD**
Операнды:	Нет
Назначение:	Чтение двойного слова из строки
Процессор:	80386+
Флаги:	Не изменяются
Комментарий:	Команда LODSD копирует двойное слово из памяти по адресу DS:(E)SI в регистр EAX. После выполнения команды, регистр SI увеличивается на 4, если флаг DF = 0, или уменьшается на 4, если DF = 1. Если команда используется в 64-разрядном режиме адресации, то используется регистр (R|E)SI.

1. Цепочечные команды

```
void *memset(void *s, int c, size_t n);
```

Функция memset() заполняет первые n байтов той области памяти, на которую указывает s, постоянным байтом c.

```
void *memset(void *s, int c, size_t n);
{
    for (size_t i = 0; i < n; ++i )
       *(s + i) = c;

    return s;
}
```

1. Цепочечные команды

void *memset (void *s, int c, size_t n);

```
section .data
    dest  db "hello"
    n     db  $ - src
    c     db  0

section .text

    xor eax, eax
    mov ecx, 0
    mov edi, dest
next:
    mov al, [c]
    mov [edi], al    ; *(dest+i) = c
    inc edi
    inc ecx          ; i++
    cmp ecx, [n]      ; strlen()
    jne next         ; jle next
```

```
void *memset(void *s, int c, size_t n);
{
    for (size_t i = 0; i < n; ++i )
       *(s + i) = c;

    return s;
}
```

1. Цепочечные команды

void *memset (void *s, int c, size_t n);

```
section .data
    dest  db "hello"
    n     db  $ - src
    c     db  0

section .text

    xor eax, eax
    mov ecx, 0
    mov edi, dest
next:
    mov al, [c]
    mov [edi + ecx], al      ; *(dest+i) = c
    inc ecx          ; i++
    cmp ecx, [n]      ; strlen()
    jne next         ; jle next
```

```
void *memset(void *s, int c, size_t n);
{
    for (size_t i = 0; i < n; ++i )
       *(s + i) = c;

    return s;
}
```

1. Цепочечные команды

Синтаксис:	**STOSB**
Операнды:	Нет
Назначение:	Запись байта в строку
Процессор:	8086+
Флаги:	Не изменяются
Комментарий:	Команда STOSB сохраняет регистр AL в ячейке памяти по адресу ES:(E)DI. После выполнения команды, регистр DI увеличивается на 1, если флаг DF = 0, или уменьшается на 1, если DF = 1. Если команда используется в 64-разрядном режиме адресации, то используется регистр (R|E)DI.

1. Цепочечные команды

void *memset (void *s, int c, size_t n);

```
section .data
    dest  db "hello"
    n     db  $ - src
    c     db  0

section .text

    xor eax, eax
    mov ecx, 0
    mov edi, dest
    cld                 ; направление
next:
    mov al, [c]
    stosb               ; *(dest+i) = c
    inc ecx             ; i++
    cmp ecx, [n]         ; strlen()
    jne next            ; jle next
```

```
void *memset(void *s, int c, size_t n);
{
    for (size_t i = 0; i < n; ++i )
       *(s + i) = c;

    return s;
}
```

1. Цепочечные команды

:: toUpper (char *src, size_t n);

```
section .data
    src  db "hello"
    n    db  $ - src

section .text

    xor eax, eax
    mov ecx, [n]  ; strlen("hello")
    mov edi, dest
    mov esi, dest
    cld
step:
    lodsb             ; al = *(src + i)
    and al, 0xDF      ;
    mov [edi], al
    inc edi           ;

    dec ecx
    cmp ecx, 0
    jne step
```

```
section .data
    src  db "hello"
    n    db  $ - src

section .text

    xor eax, eax
    mov ecx, [n]  ; strlen("hello")
    mov edi, dest
    mov esi, dest
    cld
step:
    lodsb             ; al = *(src + i)
    and al, 0xDF
    stosb             ; mov [edi], al
                      ; inc edi
    dec ecx
    cmp ecx, 0
    jne step
```

1. Цепочечные команды

:: toLower (char *src, size_t n);

```
section .data
    src  db "HELLO"
    n    db  $ - src

section .text

    xor eax, eax
    mov ecx, [n]  ; strlen("HELLO")
    mov edi, dest
    mov esi, dest
    cld
step:
    lodsb              ; al = *(src + i)
    or al, 0x20
    stosb              ; mov [edi], al
                       ; inc edi
    dec ecx
    cmp ecx, 0
    jne step
```

1. Цепочечные команды

Синтаксис:	**STOSW**
Операнды:	Нет
Назначение:	Запись слова в строку
Процессор:	8086+
Флаги:	Не изменяются
Комментарий:	Команда STOSW сохраняет регистр AX в ячейке памяти по адресу ES:(E)DI. После выполнения команды, регистр DI увеличивается на 2, если флаг DF = 0, или уменьшается на 2, если DF = 1. Если команда используется в 64-разрядном режиме адресации, то используется регистр (R|E)DI.

1. Цепочечные команды

Синтаксис:	**STOSD**
Операнды:	Нет
Назначение:	Запись двойного слова в строку
Процессор:	8086+
Флаги:	Не изменяются
Комментарий:	Команда STOSD сохраняет регистр EAX в ячейке памяти по адресу ES:(E)DI. После выполнения команды, регистр DI увеличивается на 4, если флаг DF = 0, или уменьшается на 4, если DF = 1. Если команда используется в 64-разрядном режиме адресации, то используется регистр (R|E)DI.

1. Цепочечные команды

:: toLower (char *src, size_t n);

```
; szStr  db "HeLLo"

section .text
        xor eax, eax
        mov     esi, szStr
        mov     edi, esi
        cld
.loc_loop:
        lodsb
        cmp     al,'A'
        jb      @f                  ; до буквы A
        cmp     al,'Z'
        ja      @f                  ; после Z
        or      al,0x20
@f:
        stosb
        or      al,al
        jnz     .loc_loop
```

1. Цепочечные команды

:: toUpper (char *src, size_t n);

```
; szStr  db “HeLLo”

section .text
        xor eax, eax
        mov     esi, szStr mov     edi,esi
        cld
.loc_loop:
        lodsb
        cmp     al,‘a'
        jb      @f                  ; до буквы a
        cmp     al,‘z'
        ja      @f                  ; после z
        and     al,0xDF
@f:
        stosb
        or      al,al
        jnz     .loc_loop
```

int memcmp (const void *_s1_, const void *_s2, size_t n_);

```
int memcmp (const void *s1, const void *s2, size_t n);
{
    int i = 0;

    while (n)
    {
        if(*(s1 + i) != *(s2 + i))
        {
           i ++;
           break;
        }
       n --;
    }
    return ret;
}
```

int memcmp (const void *_s1_, const void *_s2, size_t n_);

```
        mov     esi, src    ; Указатели
        mov     edi, dest
        xor ecx, ecx                ; счетчик цикла ( с 1 до N)
        xor eax, eax                ; результат

@next:
        inc ecx
        cmp ecx, [n]
        je @end
        mov edx, [esi]
        cmp edx, [edi]        ; *(dest+i) == *(src + i)
        inc esi
        inc edi                ; влияние на флаги
        je @next
        inc eax;     ; не равны
@end:
```

Ошибка!!

int memcmp (const void *_s1_, const void *_s2, size_t n_);

```
        mov     esi, src    ; Указатели
        mov     edi, dest
        xor ecx, ecx                ; счетчик цикла ( с 1 до N)
        xor eax, eax                ; результат

@next:
        inc ecx
        cmp ecx, [n]
        je @end
        mov edx, [esi]
        cmp edx, [edi]        ; *(dest+i) == *(src + i)
        jne @notequal
        inc esi
        inc edi                ; влияние на флаги
        jmp @next
@notequal:
        inc eax;     ; не равны
@end:
```

1. Цепочечные команды

Синтаксис:	**CMPSB**
Операнды:	Нет
Назначение:	Сравнение строк байтов
Процессор:	8086+
Флаги:	Флаги OF, SF, ZF, AF, PF и CF устанавливаются в соответствии с результатом.
Комментарий:	Команда CMPSB сравнивает один байт из памяти по адресу DS:SI с байтом по адресу ES:DI. Аналогична по действию команде CMP. После выполнения команды, регистры SI и DI увеличиваются на 1, если флаг DF = 0, или уменьшаются на 1, если DF = 1. Если команда используется в 64-разрядном режиме адресации, то используются регистры (R|E)SI и (R|E)DI соответственно.

1. Цепочечные команды

int memcmp (const void *_s1_, const void *_s2, size_t n_);

```
        mov     esi, src    ; Указатели
        mov     edi, dest
        xor ecx, ecx                ; счетчик
        xor eax, eax                ; результат
        cld                  ; направление
@next:
        inc ecx
        cmp ecx, [n]
        je @end
        cmpsb          ; *(dest+i) == *(src + i)
        je @next
        inc eax;       ; не равны
@end:
```

1. Цепочечные команды

Синтаксис:	**CMPSW**
Операнды:	Нет
Назначение:	Сравнение строк слов
Процессор:	8086+
Флаги:	Флаги OF, SF, ZF, AF, PF и CF устанавливаются в соответствии с результатом.
Комментарий:	Команда CMPSW сравнивает слово из памяти по адресу DS:SI со словом по адресу ES:DI. Аналогична по действию команде CMP. После выполнения команды, регистры SI и DI увеличиваются на 2, если флаг DF = 0, или уменьшаются на 2, если DF = 1. Если команда используется в 64-разрядном режиме адресации, то используются регистры (R|E)SI и (R|E)DI соответственно.

1. Цепочечные команды

Синтаксис:	**CMPSD**
Операнды:	Нет
Назначение:	Сравнение строк байтов
Процессор:	80386+
Флаги:	Флаги OF, SF, ZF, AF, PF и CF устанавливаются в соответствии с результатом.
Комментарий:	Команда CMPSD сравнивает двойное слово из памяти по адресу DS:SI с двойным словом по адресу ES:DI. Аналогична по действию команде CMP. После выполнения команды, регистры SI и DI увеличиваются на 4, если флаг DF = 0, или уменьшаются на 4, если DF = 1. Если команда используется в 32-разрядном режиме адресации, то используются регистры (R|E)SI и (R|E)DI соответственно

1. Цепочечные команды

Префиксы повторений REPx

REP	Повторять команду , пока (E)CX != 0
REPE / REPZ	Повторять команду, пока равно (флаг ZF = 1) и (E)CX != 0
REPNE / REPNZ	Повторять команду, пока НЕ равно (флаг ZF = 0) и (E)CX != 0

Префиксы повторений REPx

REP Повторять команду , пока (E)CX != 0

```
    mov ecx, [N]
    test ecx, ecx
    je @end
@next:
    instr
    dec ecx          ;
    cmp ecx, 0       ;
    jne @next        ;
@end:
```

```
    mov ecx, [N]
    rep instr
```

1. Цепочечные команды

```
section .data
    src  dd 0x11223344
    dest dd 0

section .text

    xor eax, eax
    mov ecx, 0
    mov edi, dest
    mov esi, src
    cld                 ; направление
next:
    movsb               ; *(dest+i) = *(src + i)
    inc ecx             ; i++
    cmp ecx, 4          ; sizeof(dd) == 4
    jne next            ; jle next
```

```
section .data
    src  dd 0x11223344
    dest dd 0

section .text

    xor eax, eax
    mov ecx, 4
    mov edi, dest
    mov esi, src

    cld
    rep movsb
                    ; !!! ecx == 0
```

REP Повторять команду , пока (E)CX != 0

1. Цепочечные команды

void *memcpy(void *lpDst, const void *lpSrc, size_t dSize);

```
section .text

        xor eax, eax
        cld
        mov     edi, lpDst
        mov     esi, lpSrc

        mov     ecx, [dSize]
        push    ecx
        ; Разделить на 4 и получить длину в DWORD
        shr     ecx, 2
        ; Скопировать основную часть строки DWORD'ами
        rep     movsd
        pop     ecx
        ; Получить остаток от деления на 4
        and     ecx, 3
        ; Скопировать остаток строки байтами
        rep     movsb
```

1. Цепочечные команды

void *memset (void *s, int c, size_t n);

```
section .data
    dest  db "hello"
    n     db  $ - src
    c     db  0

section .text

    xor eax, eax
    mov ecx, 0
    mov edi, dest
    cld                 ; направление
next:
    mov al, [c]
    stosb               ; *(dest+i) = c
    inc ecx             ; i++
    cmp ecx, [n]        ; strlen()
    jne next            ; jle next
```

```
section .data
    dest  db "hello"
    n     db  $ - src
    c     db  0

section .text

    xor eax, eax
    mov ecx, [n]
    mov edi, dest
    cld

    mov al, [c]
    rep stosb
```

REP Повторять команду , пока (E)CX != 0

1. Цепочечные команды

REPE / REPZ Повторять команду, пока равно (флаг ZF = 1) и (E)CX != 0

```
    mov ecx, [N]
    test ecx, ecx
    je @end
@next:
    instr
    jz @setZ
    mov [set_z], 0
    jmp @cont
@setZ
    mov [set_z], 1
@cont
    dec ecx          ;
    cmp ecx, 0       ;
    jne @next        ;
    cmp [set_z], 1
    jz @next
@end:
```

```
    mov ecx, [N]
    repz instr
```

1. Цепочечные команды

int memcmp (const void *_s1_, const void *_s2, size_t n_);

```
        mov     esi, src     ; Указатели
        mov     edi, dest
        xor ecx, ecx                ; счетчик
        xor eax, eax                ; результат
        cld                    ; направление
@next:
        inc ecx
        cmp ecx, [n]
        je @end
        cmpsb          ;
        je @next
        inc eax;     ; не равны
@end:
```

REPE / REPZ

Повторять команду, пока равно (флаг ZF = 1) и (E)CX != 0

```
        mov     esi, src
        mov     edi, dest
        mov ecx, [n]
        xor eax, eax
        cld
        repz cmpsb
        jz @end
                    ; отличие в символе (zf= 0)
        inc eax;     ; не равны
@end:
                     ; вышли по ecx = 0
```

1. Цепочечные команды

REPNE / REPNZ Повторять команду, пока НЕ равно (флаг ZF = 0) и (E)CX != 0

```
    mov ecx, [N]
    test ecx, ecx
    je @end
@next:
    instr
    jz @setZ
    mov [set_z], 0
    jmp @cont
@setZ
    mov [set_z], 1
@cont
    dec ecx          ;
    cmp ecx, 0       ;
    jne @next        ;
    cmp [set_z], 1
    jnz @next
@end:
```

```
    mov ecx, [N]
    repnz instr
```

1. Цепочечные команды

size_t strlen (const char *s);

```
section .text

        cld
        mov     edi, lpStr
        xor     ecx, ecx
        dec     ecx         ; чтобы repne отработал
        xor     eax, eax    ; al == 0, сравниваем с нулем, конец строки
        repne   scasb       ; выполнится на 1 раз больше, т.к. проверит и 0
        not     ecx         ; ecx - в дополнительном коде
                            ; перевод из него
        dec     ecx         ;
```

Спасибо