First String Instructions

BASIC ASSEMBLY

Assembly language programming
By xorpd

xorpd.net

Objectives

- We will study the three instructions:
  - STOS
  - LODS
  - MOVS
- We will learn about the Direction Flag and its significance.

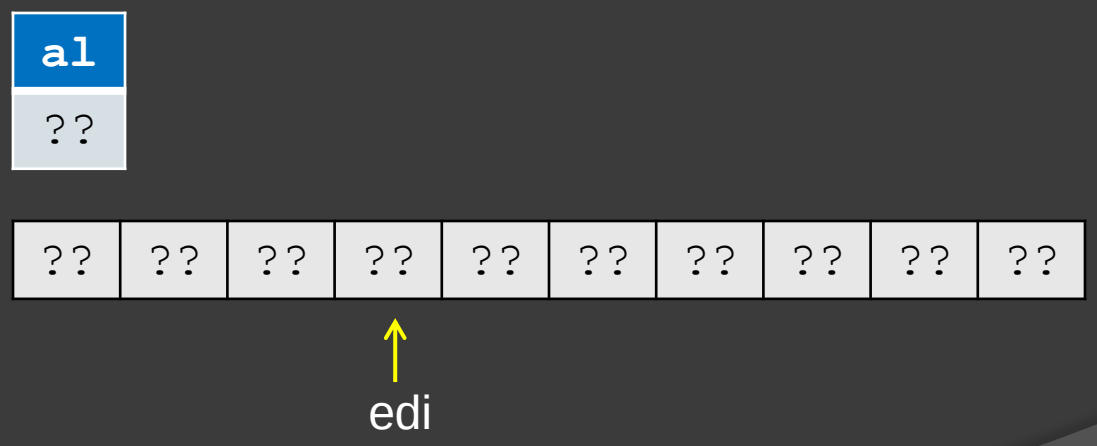
STOS

- Store String.
- Few forms:
  - STOSB (Byte)
    - $[edi] \leftarrow al$
    - edi is advanced 1 byte. (According to DF)
  - STOSW (Word)
    - $[edi] \leftarrow ax$
    - edi is advanced 2 bytes. (According to DF)
  - STOSD (Dword)
    - $[edi] \leftarrow eax$
    - edi is advanced 4 bytes. (According to DF)

STOS (Cont.)

- Example:

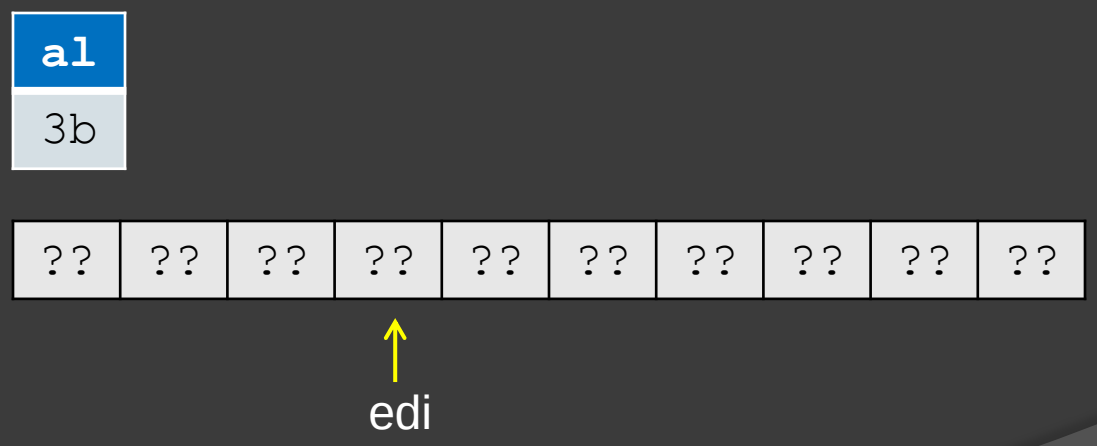
```
mov     al,3bh
stosb
```



STOS (Cont.)

- Example:

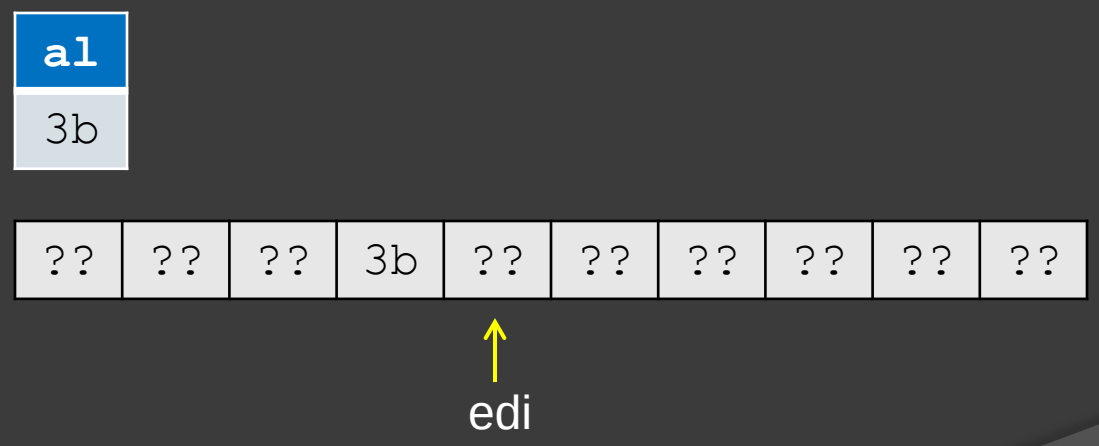
```
mov     al,3bh
stosb
```



STOS (Cont.)

- Example:

```
mov     al,3bh
stosb
```



Direction Flag

Bit number	Short name	Description
0	CF	Carry flag
1	1	Reserved
2	PF	Parity flag
3	0	Reserved
4	AF	Auxiliary Carry flag
5	0	Reserved
6	ZF	Zero flag
7	SF	Sign flag
8	TF	Trap flag
9	IF	Interrupt enable flag
10	DF	Direction Flag
11	OF	Overflow flag
More bits …		

Direction Flag (Cont.)

- The direction flag (DF) determines the **direction** for string instructions.
  - DF = 0: pointers increase.
  - DF = 1: pointers decrease.
- Changing the direction flag:
  - CLD – Clears the direction flag. (0)
  - STD – Sets the direction flag. (1)
- In your programs, the DF will usually be cleared.

Direction Flag (Example)

```
mov     eax,1fh
cld
stosd
stosd
mov     ax,2255h
std
stosw
stosw
```

Direction Flag (Example)

```
mov     eax,1fh
cld
stosd
stosd
mov     ax,2255h
std
stosw
stosw
```

edi	eax
00402000	????????

DF
0

402000	402001	402002	402003	402004	402005	402006	402007	402008	402009
00	00	00	00	00	00	00	00	00	00

edi

Direction Flag (Example)

```
mov     eax,1fh
cld
stosd
stosd
mov     ax,2255h
std
stosw
stosw
```

edi	eax
00402000	0000001f

DF
0

402000	402001	402002	402003	402004	402005	402006	402007	402008	402009
00	00	00	00	00	00	00	00	00	00

edi

Direction Flag (Example)

```
mov     eax,1fh
cld
stosd
stosd
mov     ax,2255h
std
stosw
stosw
```

edi	eax
00402000	0000001f

DF
0

402000	402001	402002	402003	402004	402005	402006	402007	402008	402009
00	00	00	00	00	00	00	00	00	00

edi

Direction Flag (Example)

```
mov     eax,1fh
cld
stosd
stosd
mov     ax,2255h
std
stosw
stosw
```

edi	eax
00402004	0000001f

DF
0

402000	402001	402002	402003	402004	402005	402006	402007	402008	402009
1f	00	00	00	00	00	00	00	00	00

edi

Direction Flag (Example)

```
mov     eax,1fh
cld
stosd
stosd
mov     ax,2255h
std
stosw
stosw
```

edi	eax
00402008	0000001f

DF
0

402000	402001	402002	402003	402004	402005	402006	402007	402008	402009
1f	00	00	00	1f	00	00	00	00	00

edi

Direction Flag (Example)

```
mov     eax,1fh
cld
stosd
stosd
mov     ax,2255h
std
stosw
stosw
```

edi	eax
00402008	00002255

DF
0

402000	402001	402002	402003	402004	402005	402006	402007	402008	402009
1f	00	00	00	1f	00	00	00	00	00

edi

Direction Flag (Example)

```
mov     eax,1fh
cld
stosd
stosd
mov     ax,2255h
std
stosw
stosw
```

edi	eax
00402008	00002255

DF
1

402000	402001	402002	402003	402004	402005	402006	402007	402008	402009
1f	00	00	00	1f	00	00	00	00	00

edi

Direction Flag (Example)

```
mov     eax,1fh
cld
stosd
stosd
mov     ax,2255h
std
stosw
stosw
```

edi	eax
00402006	00002255

DF
1

402000	402001	402002	402003	402004	402005	402006	402007	402008	402009
1f	00	00	00	1f	00	00	00	55	22

edi

Direction Flag (Example)

```
mov     eax,1fh
cld
stosd
stosd
mov     ax,2255h
std
stosw
stosw
```

edi	eax
00402004	00002255

DF
1

402000	402001	402002	402003	402004	402005	402006	402007	402008	402009
1f	00	00	00	1f	00	55	22	55	22

edi

Direction Flag – Responsible use

- The DF affects the behavior of some instructions.
- Many subroutines and pieces of code assume implicitly that DF = 0.
- Leaving the DF with the value 1 is asking for trouble.
- If you decide to set the DF, make sure to clear it later.
  - It is your responsibility.

LODS

- Load String.
- Few forms:
  - LODSB (Byte)
    - $al \leftarrow [esi]$
    - esi is advanced 1 byte. (According to DF)
  - LODSW (Word)
    - $ax \leftarrow [esi]$
    - esi is advanced 2 bytes. (According to DF)
  - LODSD (Dword)
    - $eax \leftarrow [esi]$
    - esi is advanced 4 bytes. (According to DF)

LODS (Example)

```
my_arr    dd    ARR_LEN dup (?)
```

```
  mov    esi,my_arr
  mov    ecx,ARR_LEN
  xor    edx,edx

next_element:
  lodsd
  add    edx,eax
  loop   next_element
```

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
????????	????????	????????

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
????????	????????	????????

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

esi

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
????????	????????	00000003

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

esi

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
00000000	????????	00000003

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

esi

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
00000000	00000001	00000003

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

esi

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
00000001	00000001	00000003

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

esi

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
00000001	00000001	00000002

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

esi

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
00000001	00000003	00000002

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

esi

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
00000004	00000003	00000002

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

esi

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
00000004	00000003	00000001

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

esi

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
00000004	00000005	00000001

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

esi

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
00000009	00000005	00000001

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

esi

LODS (Example)

```
my_arr     dd     ARR_LEN dup (?)
```

```
  mov     esi,my_arr
  mov     ecx,ARR_LEN
  xor     edx,edx

next_element:
  lodsd
  add     edx,eax
  loop    next_element
```

edx	eax	ecx
00000009	00000005	00000000

DF
0

my_arr

01	00	00	00	03	00	00	00	05	00	00	00

esi

MOVS

- Move data from string to string.

- Few forms:
  - MOVSB
    - $[edi] \leftarrow [esi]$ (1 byte copy)
    - esi, edi are advanced 1 byte. (According to DF)

  - MOVSW
    - $[edi] \leftarrow [esi]$ (2 bytes copy)
    - esi, edi are advanced 2 bytes. (According to DF)

  - MOVSD
    - $[edi] \leftarrow [esi]$ (4 bytes copy)
    - esi, edi are advanced 4 bytes. (According to DF)

MOVS - Notes

- MOVS can access two memory locations at the same time!
- esi is **s**ource, edi is **d**estination.
- esi, edi are both incremented or decremented, according to the DF.
- MOVS is useful for copying data.

MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```

MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
→   mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```

ecx
????????

src_arr

11	22	33	44	55	66

dst_arr

00	00	00	00	00	00

MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
→   mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```

ecx
????????

src_arr

11	22	33	44	55	66

↑ esi

dst_arr

00	00	00	00	00	00

MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
→   mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```

ecx
????????

src_arr

11	22	33	44	55	66

↑ esi

dst_arr

00	00	00	00	00	00

↑ edi

MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
→   movsb
    loop    copy_dword
```

ecx
00000006

src_arr

11	22	33	44	55	66

↑ esi

dst_arr

00	00	00	00	00	00

↑ edi

MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
→   loop    copy_dword
```

ecx
00000006

src_arr

11	22	33	44	55	66

↑ esi

dst_arr

11	00	00	00	00	00

↑ edi

MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```

ecx
00000005

src_arr

11	22	33	44	55	66

esi

dst_arr

11	00	00	00	00	00

edi

MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```

ecx
00000005

src_arr

11	22	33	44	55	66

esi

dst_arr

11	22	00	00	00	00

edi

MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```

ecx
00000004

src_arr

11	22	33	44	55	66

esi

dst_arr

11	22	00	00	00	00

edi

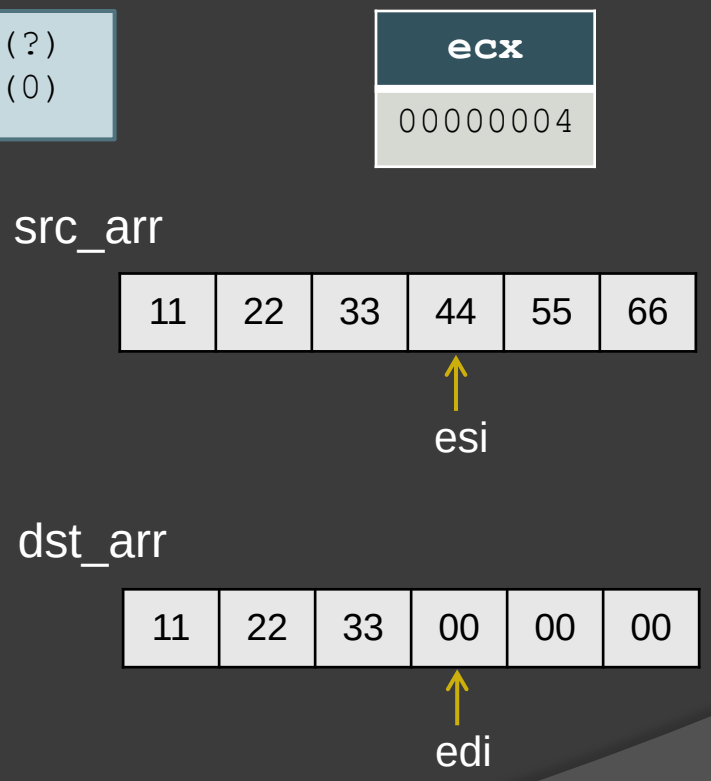
MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```



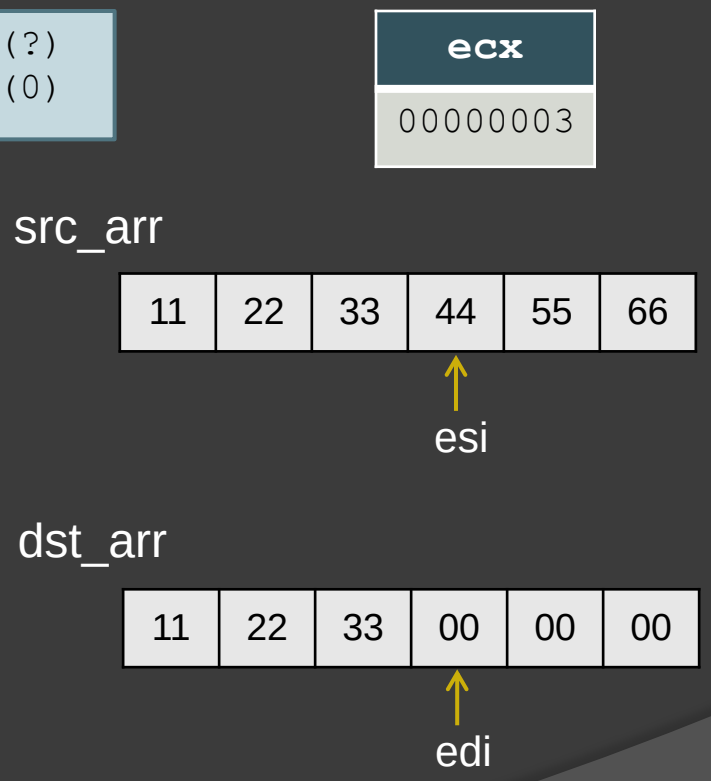
MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```



MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```

ecx
00000003

src_arr

11	22	33	44	55	66

esi

dst_arr

11	22	33	44	00	00

edi

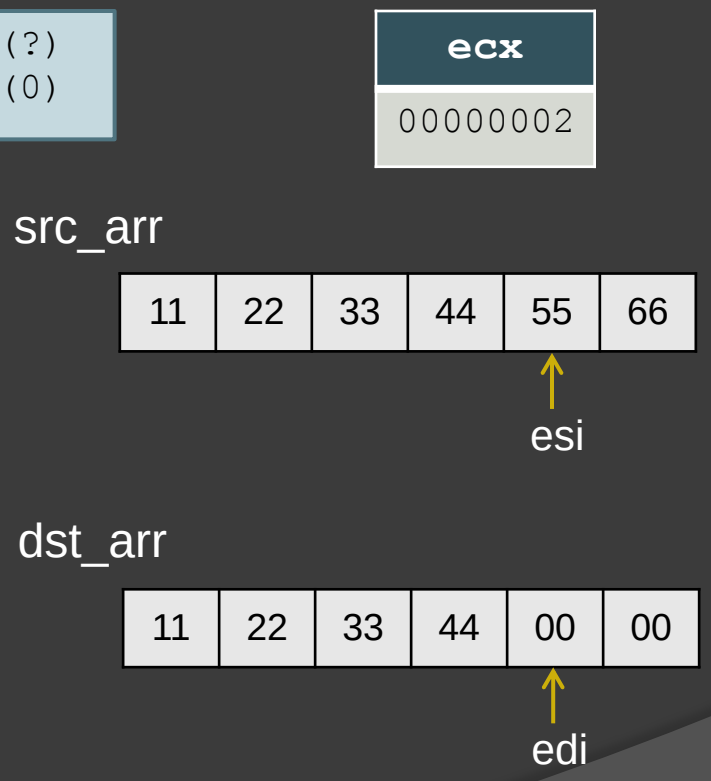
MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```



MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
→   loop    copy_dword
```

ecx
00000002

src_arr

11	22	33	44	55	66

↑ esi

dst_arr

11	22	33	44	55	00

↑ edi

MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```

ecx
00000001

src_arr

11	22	33	44	55	66

esi

dst_arr

11	22	33	44	55	00

edi

MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```

ecx
00000001

src_arr

11	22	33	44	55	66

esi

dst_arr

11	22	33	44	55	66

edi

MOVS (Example)

- Copying data

```
src_arr     db     ARR_LEN dup (?)
dst_arr     db     ARR_LEN dup (0)
```

```
    mov     esi,src_array
    mov     edi,dest_array
    mov     ecx,ARR_LEN

copy_byte:
    movsb
    loop    copy_dword
```

ecx
00000000

src_arr

11	22	33	44	55	66

esi

dst_arr

11	22	33	44	55	66

edi

Summary

- We have learned about the three instructions:
  - STOS (Store String).
  - LODS (Load string).
  - MOVS (Move data from string to string).

- The Direction Flag determines the direction to which esi or edi are advanced.
  - DF = 0: Increase.
  - DF = 1: Decrease.