Introduction to 8086 Assembly

Lecture 18

String Instructions

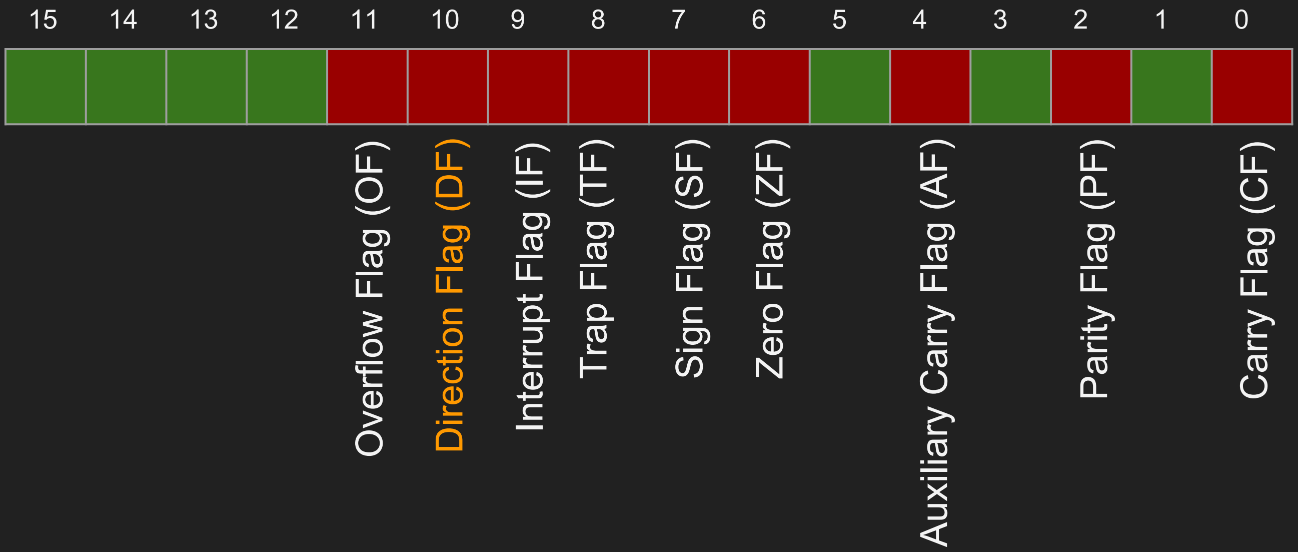
String instructions

- Working with sequence of bytes (words, double-words, quad-words)
- Using **Index** registers
  - ESI (source index)
  - EDI (destination index)

String instructions

- Working with sequence of bytes (words, double-words, quad-words)
- Using **Index** registers
  - ESI (source index)
  - EDI (destination index)
- The direction flag
  - CLD (sets DF=0)
  - STD (sets DF=1)

Remember: the FLAGS Register



```
CF: carry flag
OF: overflow flag
SF: sign flag
ZF: zero flag

PF: parity flag
DF: direction flag
IF: interrupt flag
```

Storing in a string

	DF = 0	DF = 1
STOSB	mov [EDI], AL inc EDI	mov [EDI], AL dec EDI

Storing in a string

	DF = 0	DF = 1
STOSB	mov [EDI], AL add EDI, 1	mov [EDI], AL sub EDI, 1
STOSW	mov [EDI], AX add EDI, 2	mov [EDI], AX sub EDI, 2
STOSD	mov [EDI], EAX add EDI, 4	mov [EDI], EAX sub EDI, 4

Storing in a string - 64-bit mode

K. N. Toosi
University of Technology

	DF = 0	DF = 1

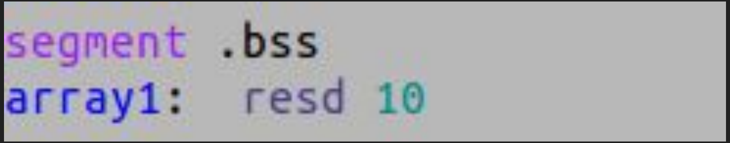
STOSB	`mov [RDI], AL`
	`add RDI, 1`
		`mov [RDI], AL`
		`sub RDI, 1`

STOSW	`mov [RDI], AX`
	`add RDI, 2`
		`mov [RDI], AX`
		`sub RDI, 2`

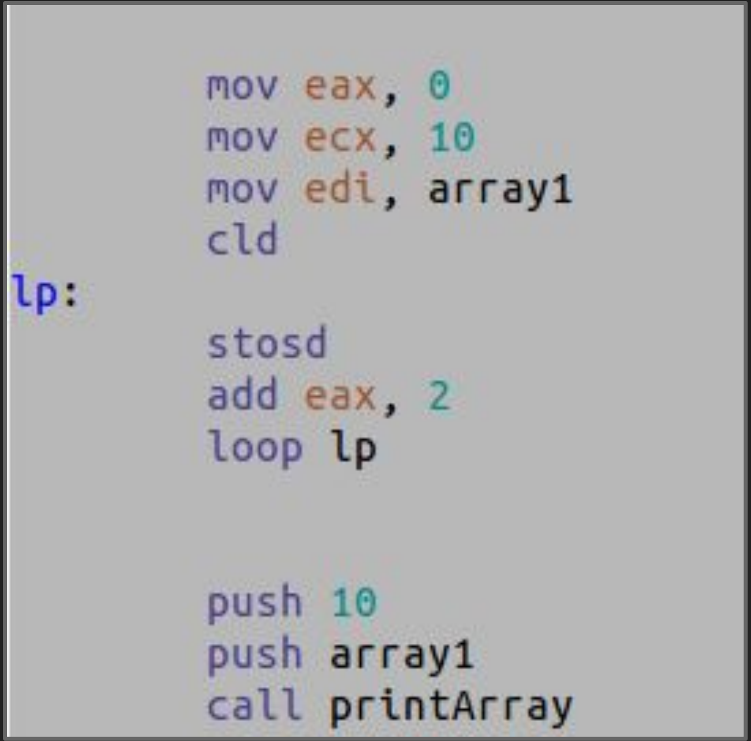
STOSD	`mov [RDI], EAX`
	`add RDI, 4`
		`mov [RDI], EAX`
		`sub RDI, 4`

STOSQ	`mov [RDI], RAX`
	`add RDI, 8`
		`mov [RDI], RAX`
		`sub RDI, 8`

Example

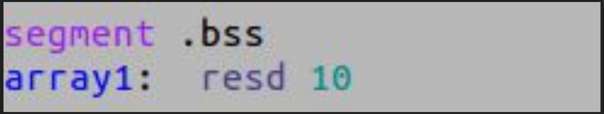


```
segment .bss
array1:  resd 10
```

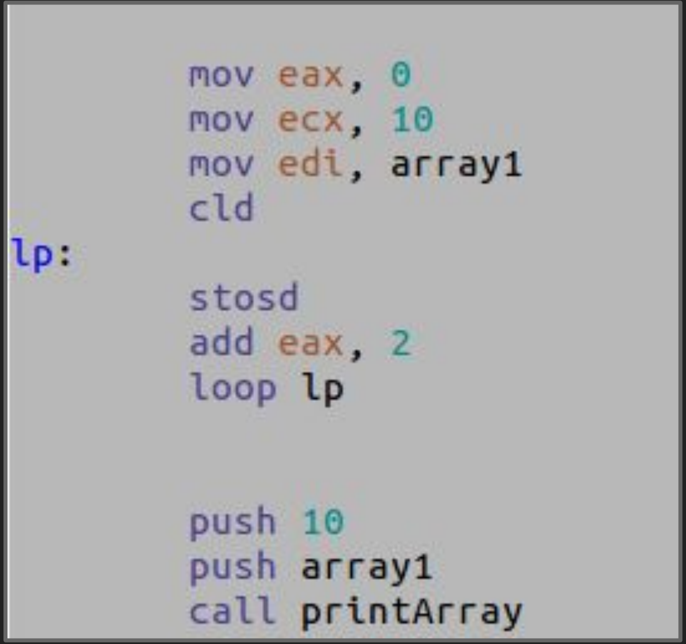


```
        mov eax, 0
        mov ecx, 10
        mov edi, array1
        cld
lp:
        stosd
        add eax, 2
        loop lp


        push 10
        push array1
        call printArray
```

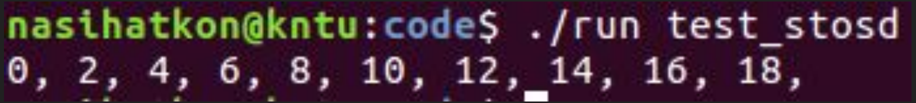
Example



```
segment .bss
array1:  resd 10
```



```
        mov eax, 0
        mov ecx, 10
        mov edi, array1
        cld
lp:
        stosd
        add eax, 2
        loop lp


        push 10
        push array1
        call printArray
```



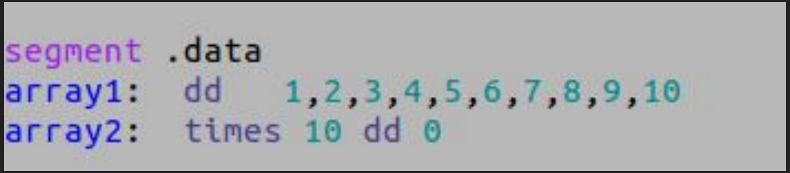
```
nasihatkon@kntu:code$ ./run test_stosd
0, 2, 4, 6, 8, 10, 12, 14, 16, 18,
```

Reading a string

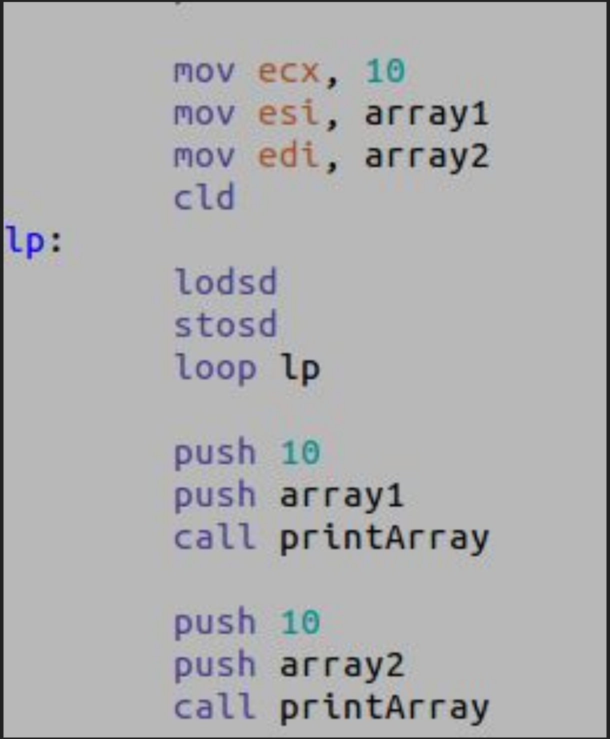
	DF = 0	DF = 1
LODSB	mov AL, [ESI] add ESI, 1	mov AL, [ESI] sub ESI, 1
LODSW	mov AX, [ESI] add ESI, 2	mov AX, [ESI] sub ESI, 2
LODSD	mov EAX, [ESI] add ESI, 4	mov EAX, [ESI] sub ESI, 4

K. N. Toosi
University of Technology

Reading a string

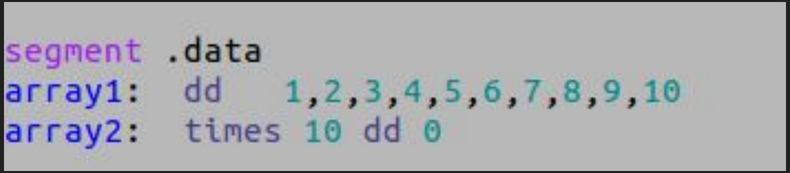


```
segment .data
array1:  dd   1,2,3,4,5,6,7,8,9,10
array2:  times 10 dd 0
```

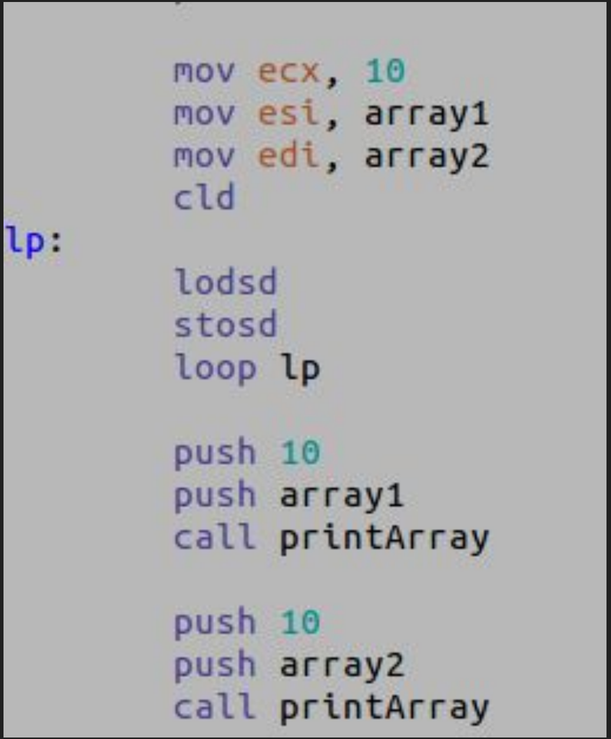


```
        mov ecx, 10
        mov esi, array1
        mov edi, array2
        cld
lp:
        lodsd
        stosd
        loop lp

        push 10
        push array1
        call printArray

        push 10
        push array2
        call printArray
```

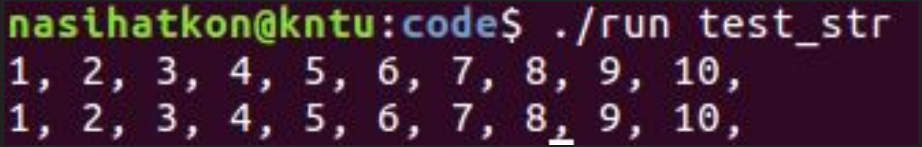
Reading a string



```
segment .data
array1:  dd   1,2,3,4,5,6,7,8,9,10
array2:  times 10 dd 0
```



```
        mov ecx, 10
        mov esi, array1
        mov edi, array2
        cld
lp:
        lodsd
        stosd
        loop lp

        push 10
        push array1
        call printArray

        push 10
        push array2
        call printArray
```



```
nasihatkon@kntu:code$ ./run test_str
1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
```

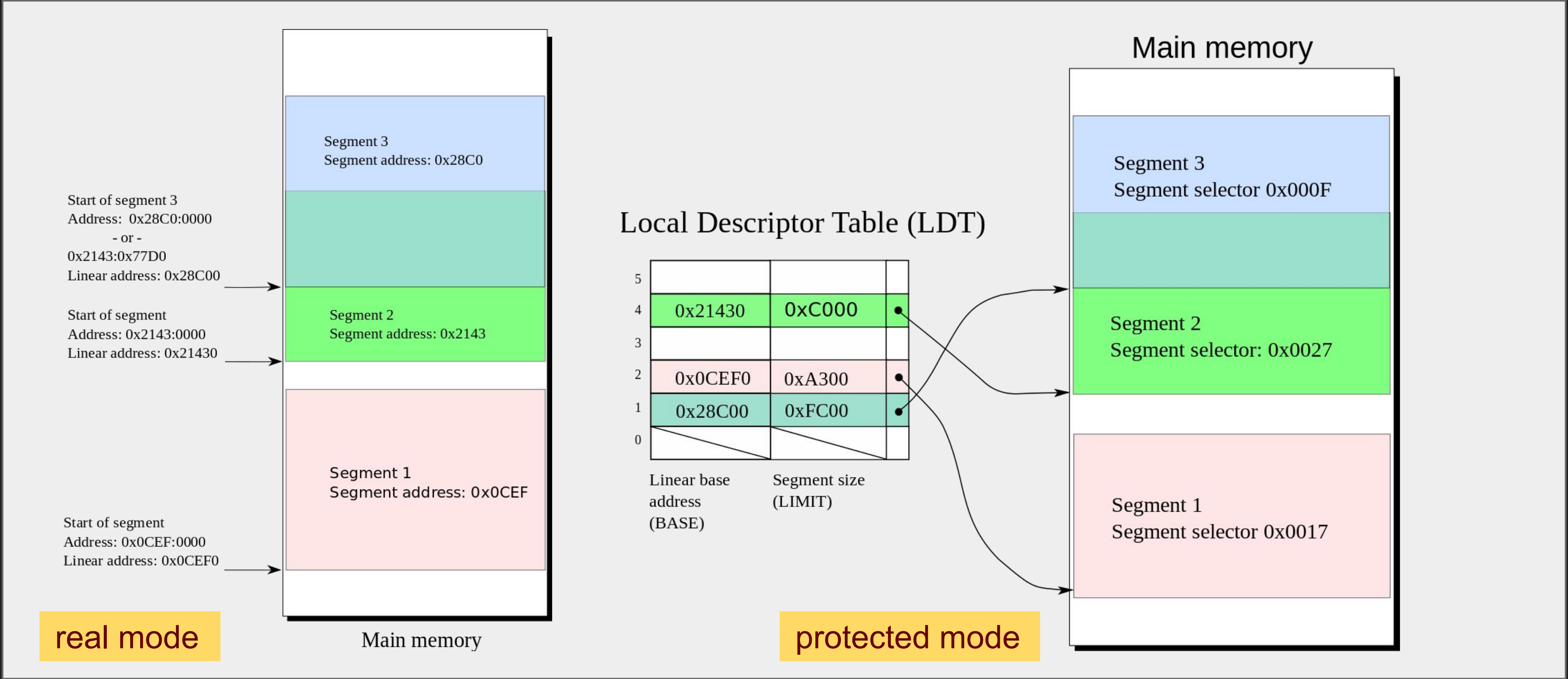
The full story!

K. N. Toosi
University of Technology

	DF = 0	DF = 1
STOSB	mov [ES:EDI], AL add EDI, 1	mov [ES:EDI], AL sub EDI, 1
STOSW	mov [ES:EDI], AX add EDI, 2	mov [ES:EDI], AX sub EDI, 2
STOSD	mov [ES:EDI],EAX add EDI, 4	mov [ES:EDI],EAX sub EDI, 4

	DF = 0	DF = 1
LODSB	mov AL, [DS:ESI] add ESI, 1	mov AL, [DS:ESI] sub ESI, 1
LODSW	mov AX, [DS:ESI] add ESI, 2	mov AX, [DS:ESI] sub ESI, 2
LODSD	mov EAX,[DS:ESI] add ESI, 4	mov EAX,[DS:ESI] sub ESI, 4

Segmentation



https://en.wikipedia.org/wiki/X86_memory_segmentation

The full story!

K. N. Toosi
University of Technology

	DF = 0	DF = 1
STOSB	mov [ES:EDI], AL add EDI, 1	mov [ES:EDI], AL sub EDI, 1
STOSW	mov [ES:EDI], AX add EDI, 2	mov [ES:EDI], AX sub EDI, 2
STOSD	mov [ES:EDI],EAX add EDI, 4	mov [ES:EDI],EAX sub EDI, 4

	DF = 0	DF = 1
LODSB	mov AL, [DS:ESI] add ESI, 1	mov AL, [DS:ESI] sub ESI, 1
LODSW	mov AX, [DS:ESI] add ESI, 2	mov AX, [DS:ESI] sub ESI, 2
LODSD	mov EAX,[DS:ESI] add ESI, 4	mov EAX,[DS:ESI] sub ESI, 4

string copy instructions

K. N. Toosi
University of Technology

	DF = 0	DF = 1
MOVSB	mov [EDI],[ESI] add ESI, 1 add EDI, 1	mov [EDI],[ESI] sub ESI, 1 sub EDI, 1
MOVSW	mov [EDI],[ESI] add ESI, 2 add EDI, 2	mov [EDI],[ESI] sub ESI, 2 sub EDI, 2
MOVSD	mov [EDI],[ESI] add ESI, 4 add EDI, 4	mov [EDI],[ESI] sub ESI, 4 sub EDI, 4

`mov [EDI],[ESI]` is for illustration
(`mov mem, mem` is not valid)

string copy instructions: full story

	DF = 0	DF = 1

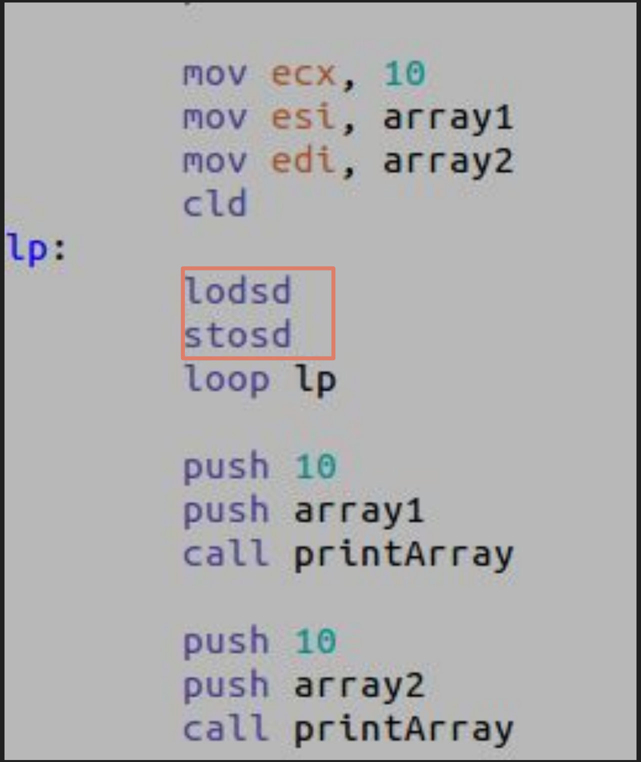
MOVSB	`mov [ES:EDI],[DS:ESI]`
	`add ESI, 1`
	`add EDI, 1`
		`mov [ES:EDI],[DS:ESI]`
		`sub ESI, 1`
		`sub EDI, 1`

MOVSW	`mov [ES:EDI],[DS:ESI]`
	`add ESI, 2`
	`add EDI, 2`
		`mov [ES:EDI],[DS:ESI]`
		`sub ESI, 2`
		`sub EDI, 2`

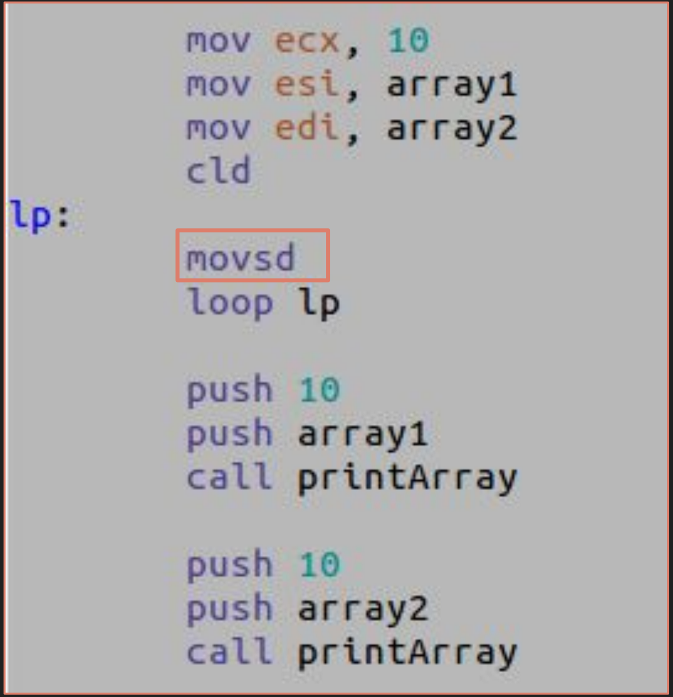
MOVSD	`mov [ES:EDI],[DS:ESI]`
	`add ESI, 4`
	`add EDI, 4`
		`mov [ES:EDI],[DS:ESI]`
		`sub ESI, 4`
		`sub EDI, 4`

`mov [ES:EDI],[DS:ESI]` is for illustration
(`mov mem, mem` is not valid)

Reading a string

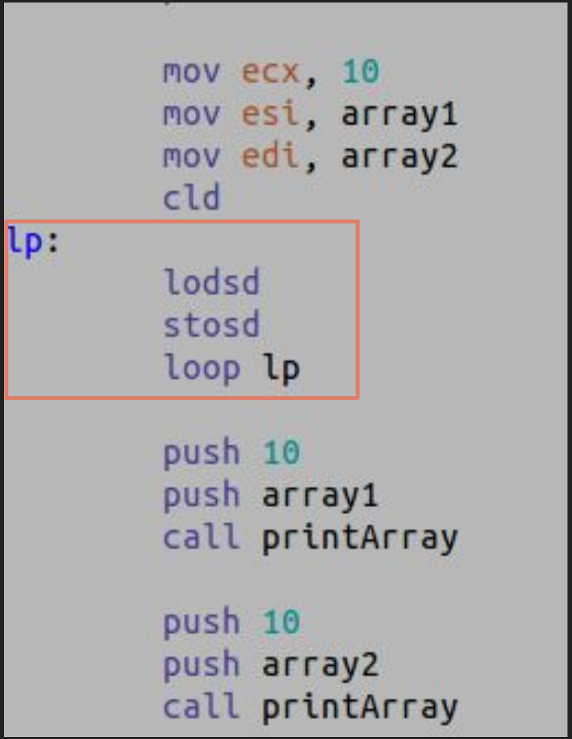


```
        mov ecx, 10
        mov esi, array1
        mov edi, array2
        cld
lp:
        lodsd
        stosd
        loop lp

        push 10
        push array1
        call printArray

        push 10
        push array2
        call printArray
```

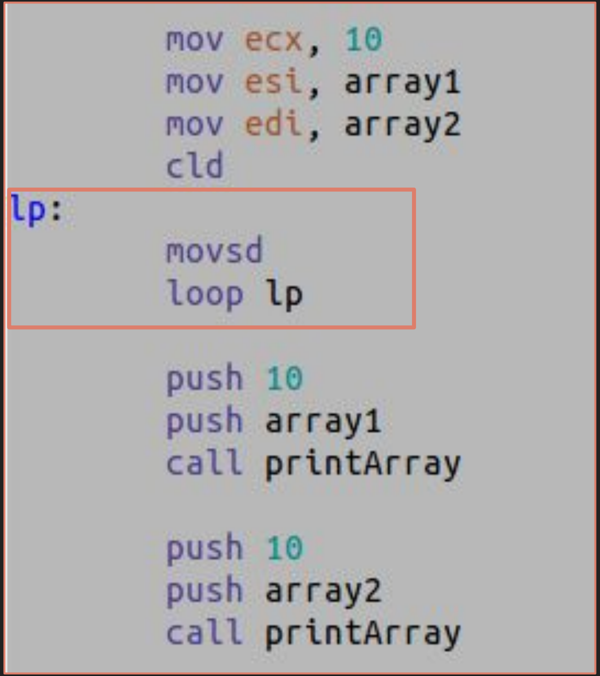


```
        mov ecx, 10
        mov esi, array1
        mov edi, array2
        cld
lp:
        movsd
        loop lp

        push 10
        push array1
        call printArray

        push 10
        push array2
        call printArray
```

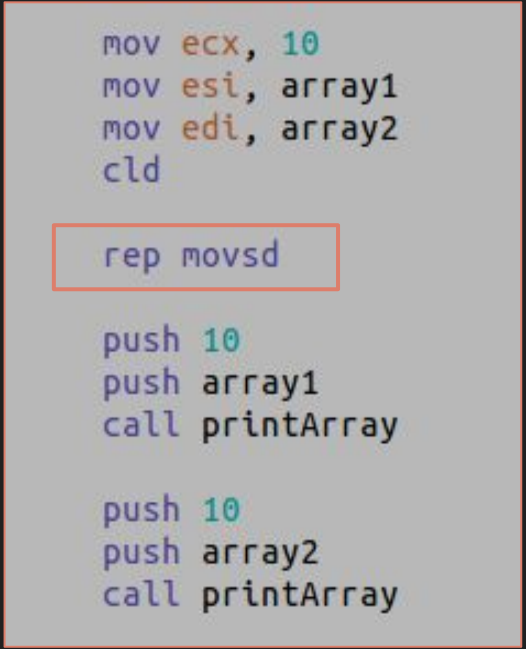
The rep instruction prefix



```
        mov ecx, 10
        mov esi, array1
        mov edi, array2
        cld
lp:
        lodsd
        stosd
        loop lp

        push 10
        push array1
        call printArray

        push 10
        push array2
        call printArray
```



```
        mov ecx, 10
        mov esi, array1
        mov edi, array2
        cld
lp:
        movsd
        loop lp

        push 10
        push array1
        call printArray

        push 10
        push array2
        call printArray
```



```
        mov ecx, 10
        mov esi, array1
        mov edi, array2
        cld

        rep movsd

        push 10
        push array1
        call printArray

        push 10
        push array2
        call printArray
```

REPx instruction prefixes

`REPE, REPZ`	(repeat while equal/zero)
`REPNE, REPNZ`	(repeat while not equal/not zero)

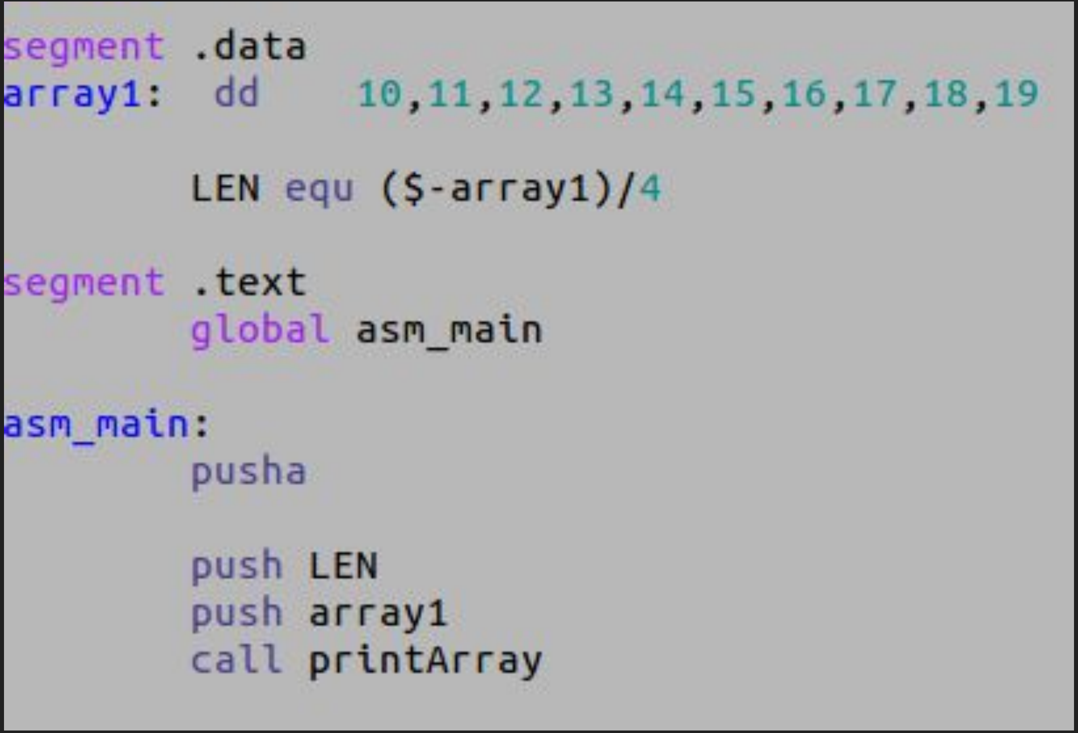
Searching strings

	DF = 0	DF = 1
SCASB	`cmp AL, [EDI]` (sets FLAGS) `add EDI, 1` (FLAGS unchanged)	`cmp AL, [EDI]` (sets FLAGS) `sub EDI, 1` (FLAGS unchanged)
SCASW	`cmp AX, [EDI]` (sets FLAGS) `add EDI, 2` (FLAGS unchanged)	`cmp AX, [EDI]` (sets FLAGS) `sub EDI, 2` (FLAGS unchanged)
SCASD	`cmp EAX,[EDI]` (sets FLAGS) `add EDI, 4` (FLAGS unchanged)	`cmp EAX,[EDI]` (sets FLAGS) `sub EDI, 4` (FLAGS unchanged)

[EDI] => [ES:EDI]

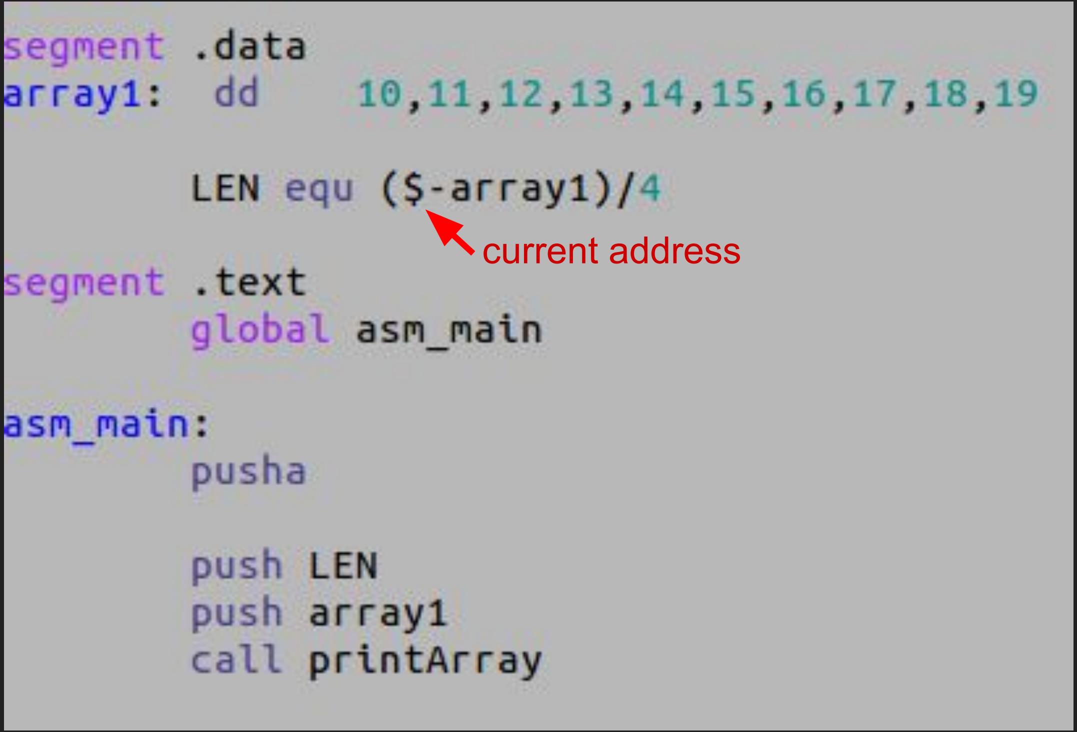
Searching for an element in array

K. N. Toosi
University of Technology



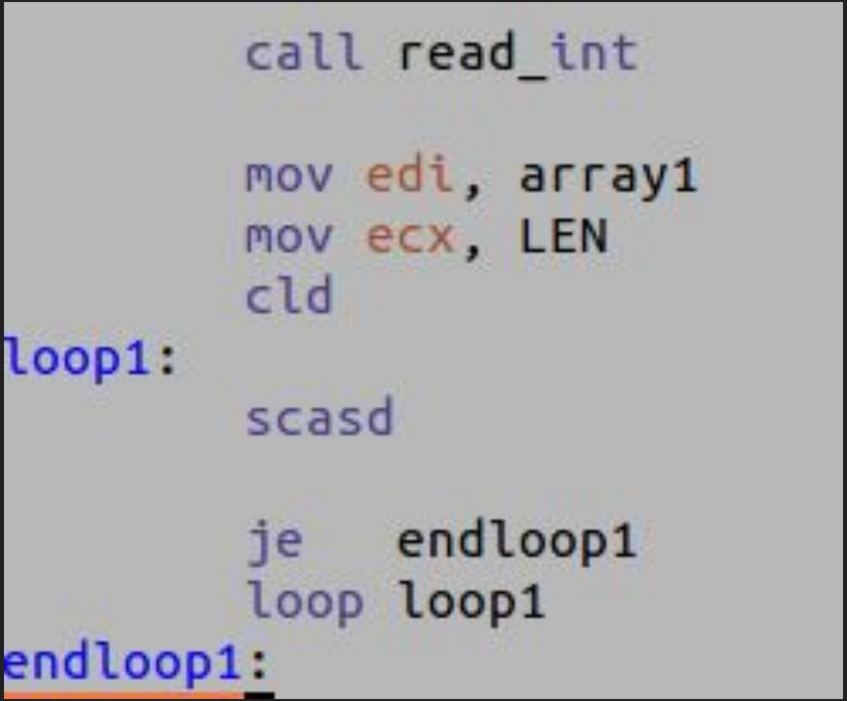
```
segment .data
array1:  dd    10,11,12,13,14,15,16,17,18,19

        LEN equ ($-array1)/4

segment .text
        global asm_main

asm_main:
        pusha

        push LEN
        push array1
        call printArray
```

Searching for an element in array



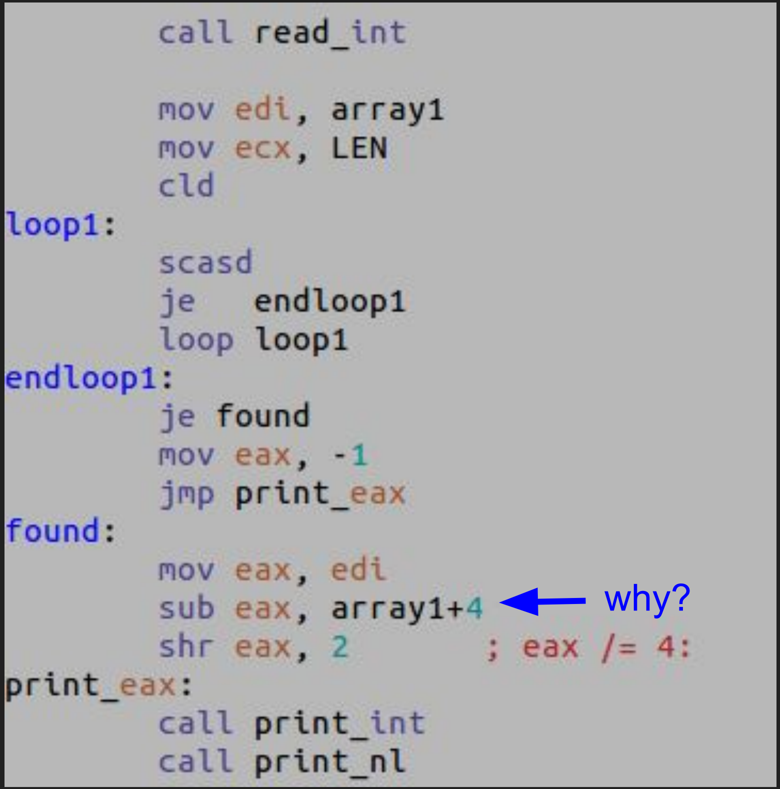
```
segment .data
array1:  dd    10,11,12,13,14,15,16,17,18,19

        LEN equ ($-array1)/4

segment .text
        global asm_main

asm_main:
        pusha

        push LEN
        push array1
        call printArray
```

Searching for an element in array



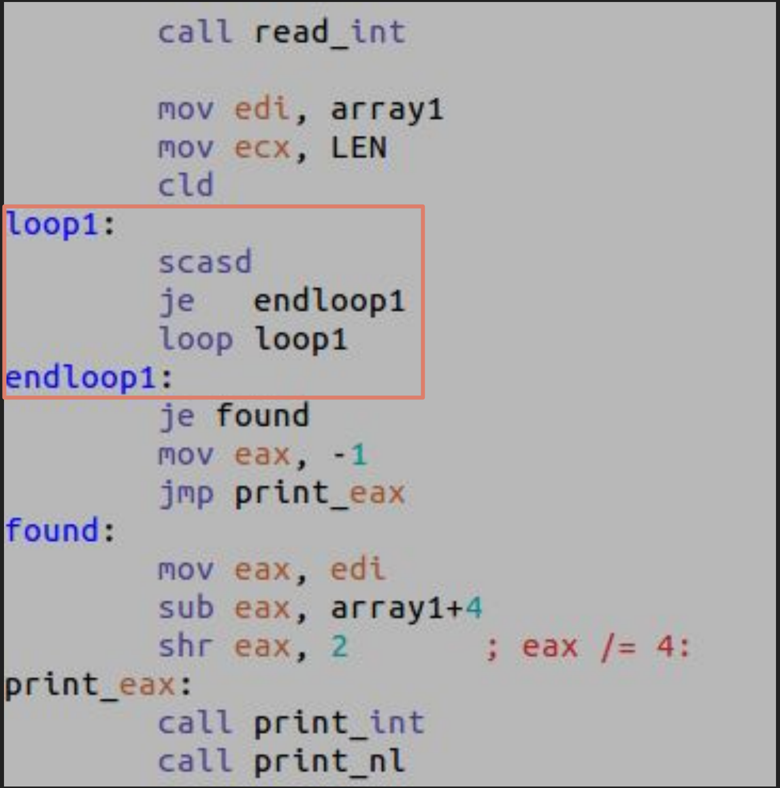
```
        call read_int

        mov edi, array1
        mov ecx, LEN
        cld
loop1:
        scasd

        je   endloop1
        loop loop1
endloop1:
```

Searching for an element in array

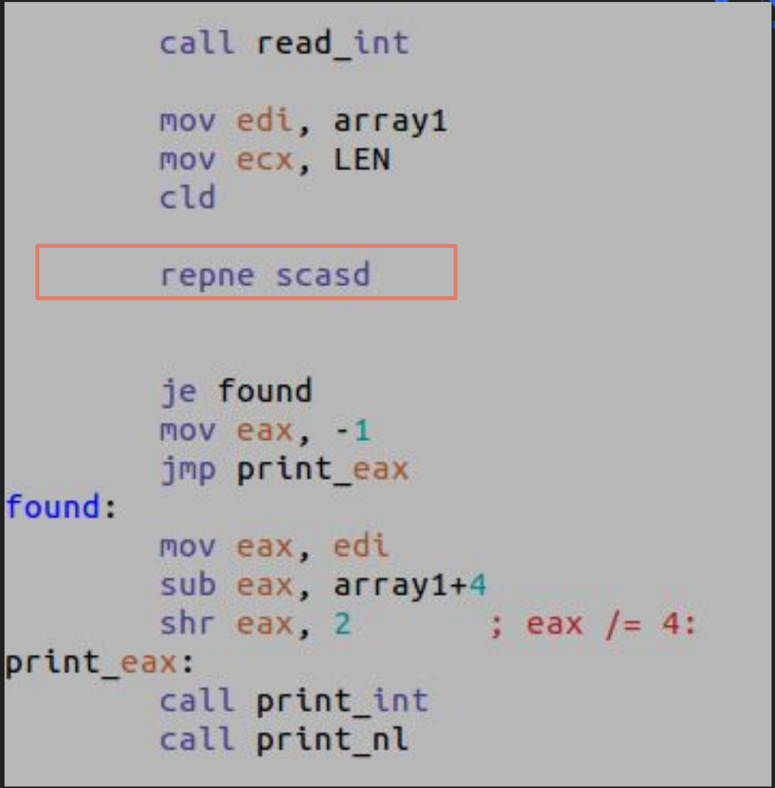


```
        call read_int

        mov edi, array1
        mov ecx, LEN
        cld
loop1:
        scasd
        je   endloop1
        loop loop1
endloop1:
        je found
        mov eax, -1
        jmp print_eax
found:
        mov eax, edi
        sub eax, array1+4
        shr eax, 2          ; eax /= 4:
print_eax:
        call print_int
        call print_nl
```

REPx instructions



```
        call read_int

        mov edi, array1
        mov ecx, LEN
        cld
loop1:
        scasd
        je   endloop1
        loop loop1
endloop1:
        je found
        mov eax, -1
        jmp print_eax
found:
        mov eax, edi
        sub eax, array1+4
        shr eax, 2          ; eax /= 4:
print_eax:
        call print_int
        call print_nl
```



```
        call read_int

        mov edi, array1
        mov ecx, LEN
        cld

        repne scasd


        je found
        mov eax, -1
        jmp print_eax
found:
        mov eax, edi
        sub eax, array1+4
        shr eax, 2          ; eax /= 4:
print_eax:
        call print_int
        call print_nl
```

Comparing strings

K. N. Toosi
University of Technology

	DF = 0	DF = 1

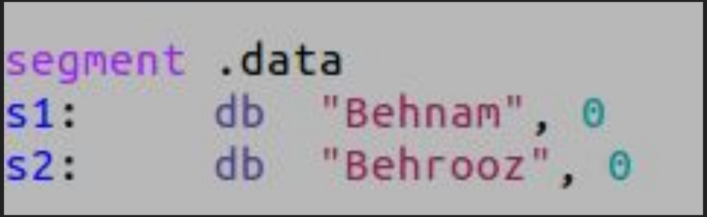
CMPSB	`cmp [EDI],[ESI]   (sets FLAGS)`
	`add ESI, 1   (FLAGS unchanged)`
	`add EDI, 1   (FLAGS unchanged)`
		`cmp [EDI],[ESI]   (sets FLAGS)`
		`sub ESI, 1   (FLAGS unchanged)`
		`sub EDI, 1   (FLAGS unchanged)`

CMPSW	`cmp [EDI],[ESI]   (sets FLAGS)`
	`add ESI, 2   (FLAGS unchanged)`
	`add EDI, 2   (FLAGS unchanged)`
		`cmp [EDI],[ESI]   (sets FLAGS)`
		`sub ESI, 2   (FLAGS unchanged)`
		`sub EDI, 2   (FLAGS unchanged)`

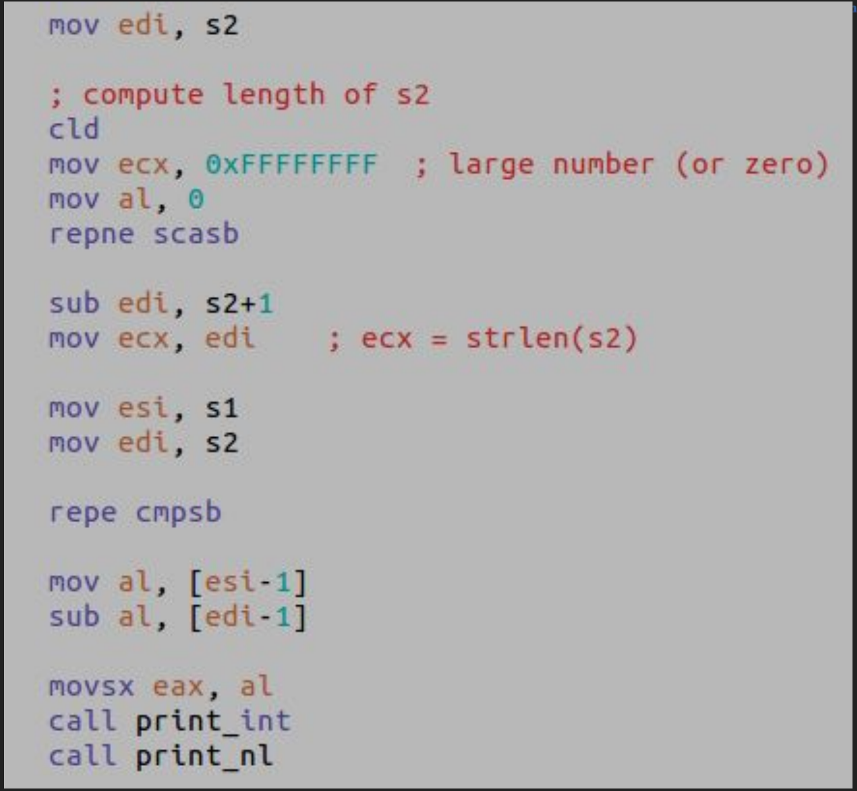
CMPSD	`cmp [EDI],[ESI]   (sets FLAGS)`
	`add ESI, 4   (FLAGS unchanged)`
	`add EDI, 4   (FLAGS unchanged)`
		`cmp [EDI],[ESI]   (sets FLAGS)`
		`sub ESI, 4   (FLAGS unchanged)`
		`sub EDI, 4   (FLAGS unchanged)`

```
[ESI] => [DS:ESI]          [EDI] => [ES:EDI]
```

Comparing strings, strcmp



```
segment .data
s1:     db  "Behnam", 0
s2:     db  "Behrooz", 0
```



```
mov edi, s2

; compute length of s2
cld
mov ecx, 0xFFFFFFFF  ; large number (or zero)
mov al, 0
repne scasb

sub edi, s2+1
mov ecx, edi    ; ecx = strlen(s2)

mov esi, s1
mov edi, s2

repe cmpsb

mov al, [esi-1]
sub al, [edi-1]

movsx eax, al
call print_int
call print_nl
```

Inline Example

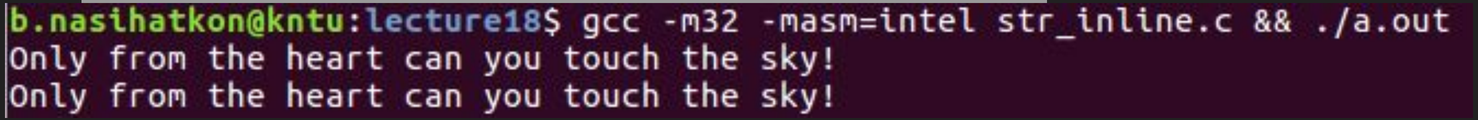
str_inline.c

```
char s1[] = "Only from the heart can you touch the sky!";
char s2[100];

int n = strlen(s1);

asm volatile ("cld;"
               "rep movsb"
              :
              : "S" (s1), "D" (s2), "c" (n+1)
              : "cc", "memory"
              );
puts(s1);
puts(s2);
```

Inline Example

str_inline.c

```c
char s1[] = "Only from the heart can you touch the sky!";
char s2[100];

int n = strlen(s1);

asm volatile ("cld;"
              "rep movsb"
              :
              : "S" (s1), "D" (s2), "c" (n+1)
              : "cc", "memory"
              );
puts(s1);
puts(s2);
```



```
b.nasihatkon@kntu:lecture18$ gcc -m32 -masm=intel str_inline.c && ./a.out
Only from the heart can you touch the sky!
Only from the heart can you touch the sky!
```