CS/EEE/INSTR F241
Lab 4 – String Operations

Anubhav Elhence

String Operations

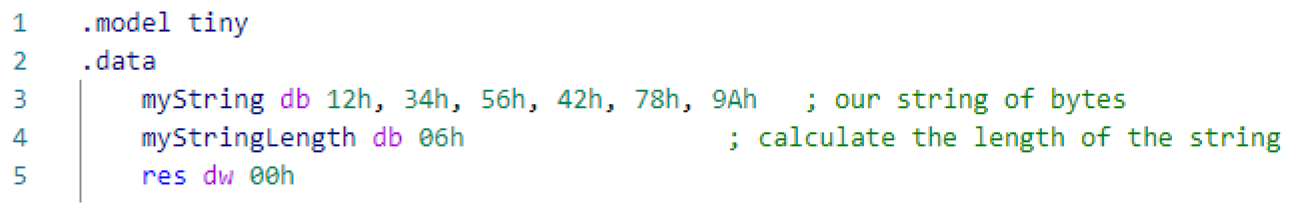
OPCODE	OPERAND	EXPLANATION	EXAMPLE
REP	instruction	repeat the given instruction till CX != 0	REP MOVSB
REPE	instruction	repeat the given instruction while CX = 0	REPE
REPZ	instruction	repeat the given instruction while ZF = 1	REPZ
REPNE	instruction	repeat the given instruction while CX != 0	REPNE
REPNZ	instruction	repeat the given instruction while ZF = 0	REPNZ
MOVSB	none	moves contents of byte given by DS:SI into ES:DI	MOVSB
MOVSW	none	moves contents of word given by DS:SI into ES:DI	MOVSW
MOVD	none	moves contents of double word given by DS:SI into ES:DI	MOVD

OPCODE	OPERAND	EXPLANATION	EXAMPLE
MOVD	none	moves contents of double word given by DS:SI into ES:DI	MOVD
LODSB	none	moves the byte at address DS:SI into AL; SI is incr/decr by 1	LODSB
LODSW	none	moves the word at address DS: SI into AX; SI is incr/decr by 2	LODSW
LODSD	none	moves the double word at address DS:SI into EAX; SI is incr/decr by 4	LODSD
STOSB	none	moves contents of AL to byte address given by ES:DI; DI is incr/dec by 1	STOSB
STOSW	none	moves the contents of AX to the word address given by ES:DI; DI is incr/decr by 2	STOSW
STOSD	none	moves contents of EAX to the DOUBLE WORD address given by ES:DI; DI is incr/decr by 4	STOSD

OPCODE	OPERAND	EXPLANATION	EXAMPLE
SCASB	none	compares byte at ES:DI with AL and sets flags according to result	SCASB
SCASW	none	compares word at ES:DI with AX and sets flags	SCASW
SCASD	none	compares double word at ES:DI with EAX and sets flags	SCASD
CMPSB	none	compares byte at ES:DI with byte at DS:SI and sets flags	CMPSB
CMPSW	none	compares word at ES:DI with word at DS:SI and sets flags	CMPSW
CMPSD	none	compares double word at ES:DI with double word at DS:SI and sets flags	CMPSD

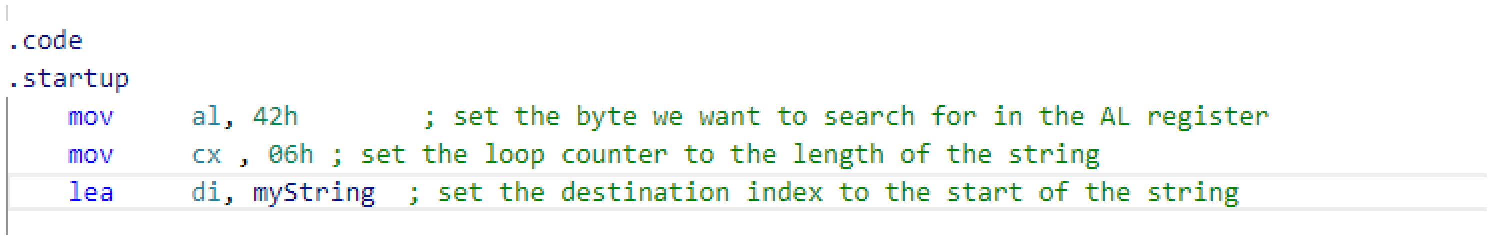
Follow Along Example

- Write an ALP to find the first occurrence of character in a string of characters. Store the Index in memory location “Res”.
- Assume the starting point-



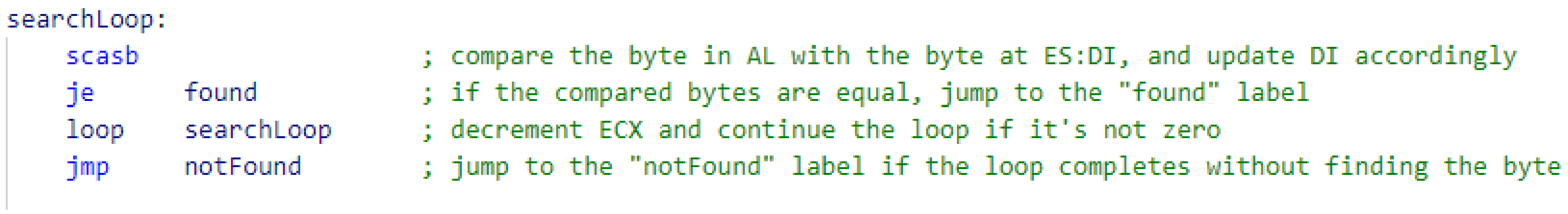
```
.model tiny
.data
    myString db 12h, 34h, 56h, 42h, 78h, 9Ah    ; our string of bytes
    myStringLength db 06h                    ; calculate the length of the string
    res dw 00h
```

- Which byte do you want to replace ? Store that in AL.
  What is the length of string ? Store than in CX.
  Which String Operation is most efficient to use in this?



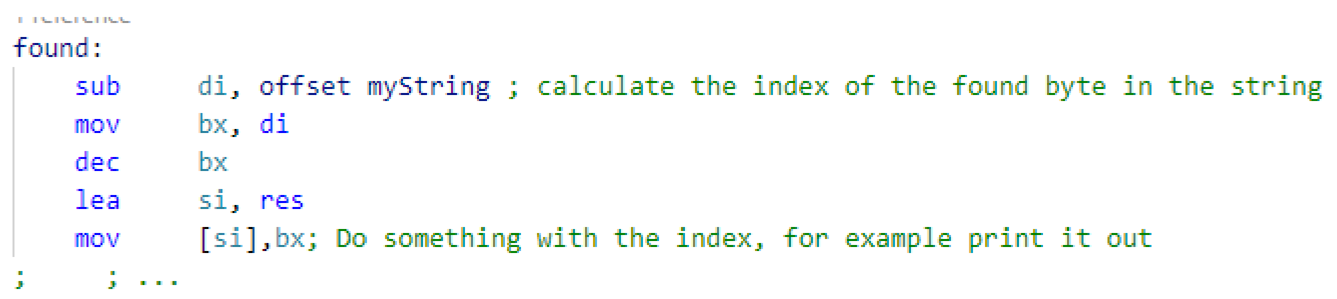
```
.code
.startup
    mov     al, 42h         ; set the byte we want to search for in the AL register
    mov     cx , 06h ; set the loop counter to the length of the string
    lea     di, myString  ; set the destination index to the start of the string
```

- Can we start the loop now?



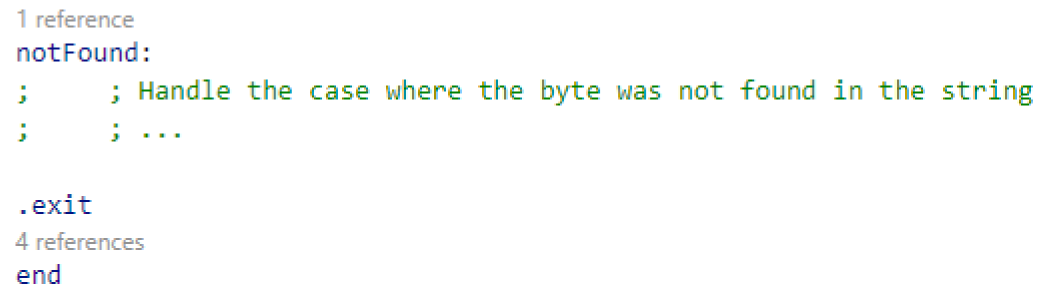
```
searchLoop:
    scasb                        ; compare the byte in AL with the byte at ES:DI, and update DI accordingly
    je      found                ; if the compared bytes are equal, jump to the "found" label
    loop    searchLoop           ; decrement ECX and continue the loop if it's not zero
    jmp     notFound             ; jump to the "notFound" label if the loop completes without finding the byte
```

What if we found the character?



```
found:
    sub     di, offset myString ; calculate the index of the found byte in the string
    mov     bx, di
    dec     bx
    lea     si, res
    mov     [si],bx; Do something with the index, for example print it out
;       ; ...
```

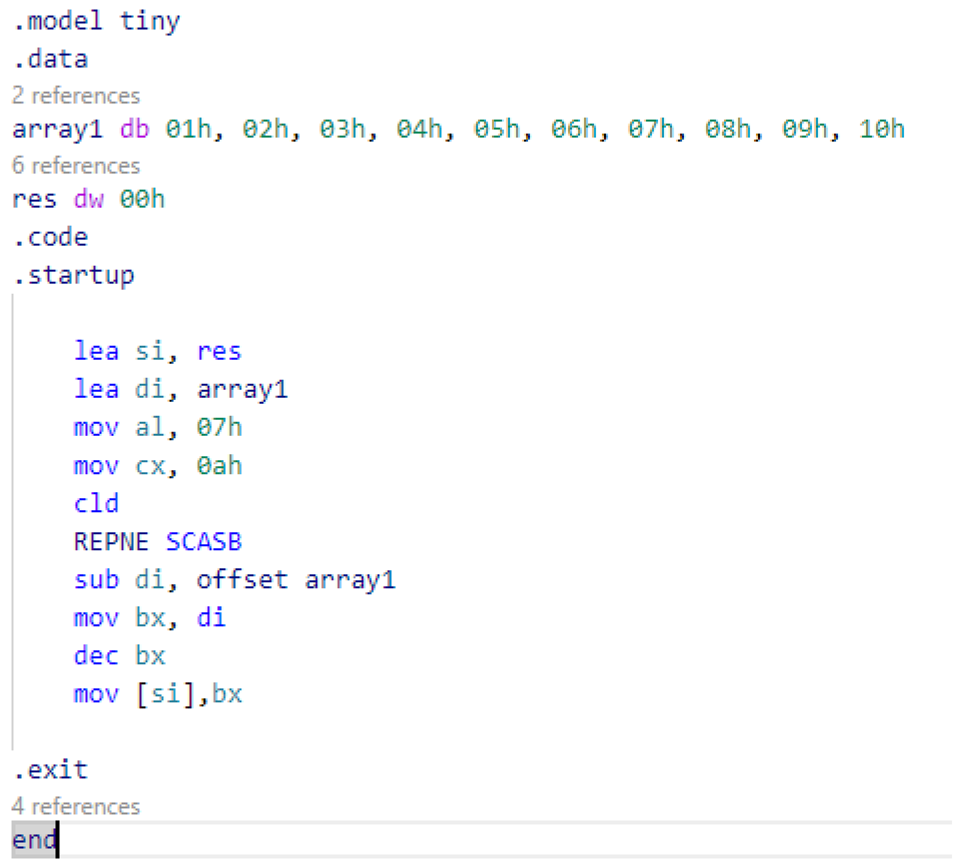
What if we did not find the character?



```
1 reference
notFound:
;       ; Handle the case where the byte was not found in the string
;       ; ...

.exit
4 references
end
```

Another Simple Way to do this.



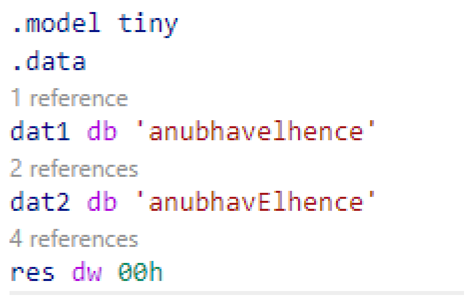
```
.model tiny
.data
2 references
array1 db 01h, 02h, 03h, 04h, 05h, 06h, 07h, 08h, 09h, 10h
6 references
res dw 00h
.code
.startup

    lea si, res
    lea di, array1
    mov al, 07h
    mov cx, 0ah
    cld
    REPNE SCASB
    sub di, offset array1
    mov bx, di
    dec bx
    mov [si],bx

.exit
4 references
end
```

Follow Along Example - 2

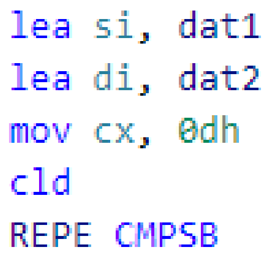
- Write an ALP to compare two strings and store the index where the two strings mismatch in memory location "RES"



```
.model tiny
.data
1 reference
dat1 db 'anubhavelhence'
2 references
dat2 db 'anubhavElhence'
4 references
res dw 00h
```

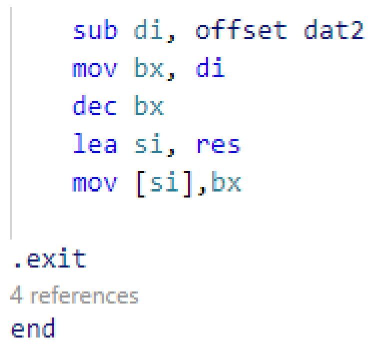
- Try on your own ! Think which String operation would best work in this situation.

- Did you figure it out ? CMPSB would work the best
- What is the string size? Store it in CX



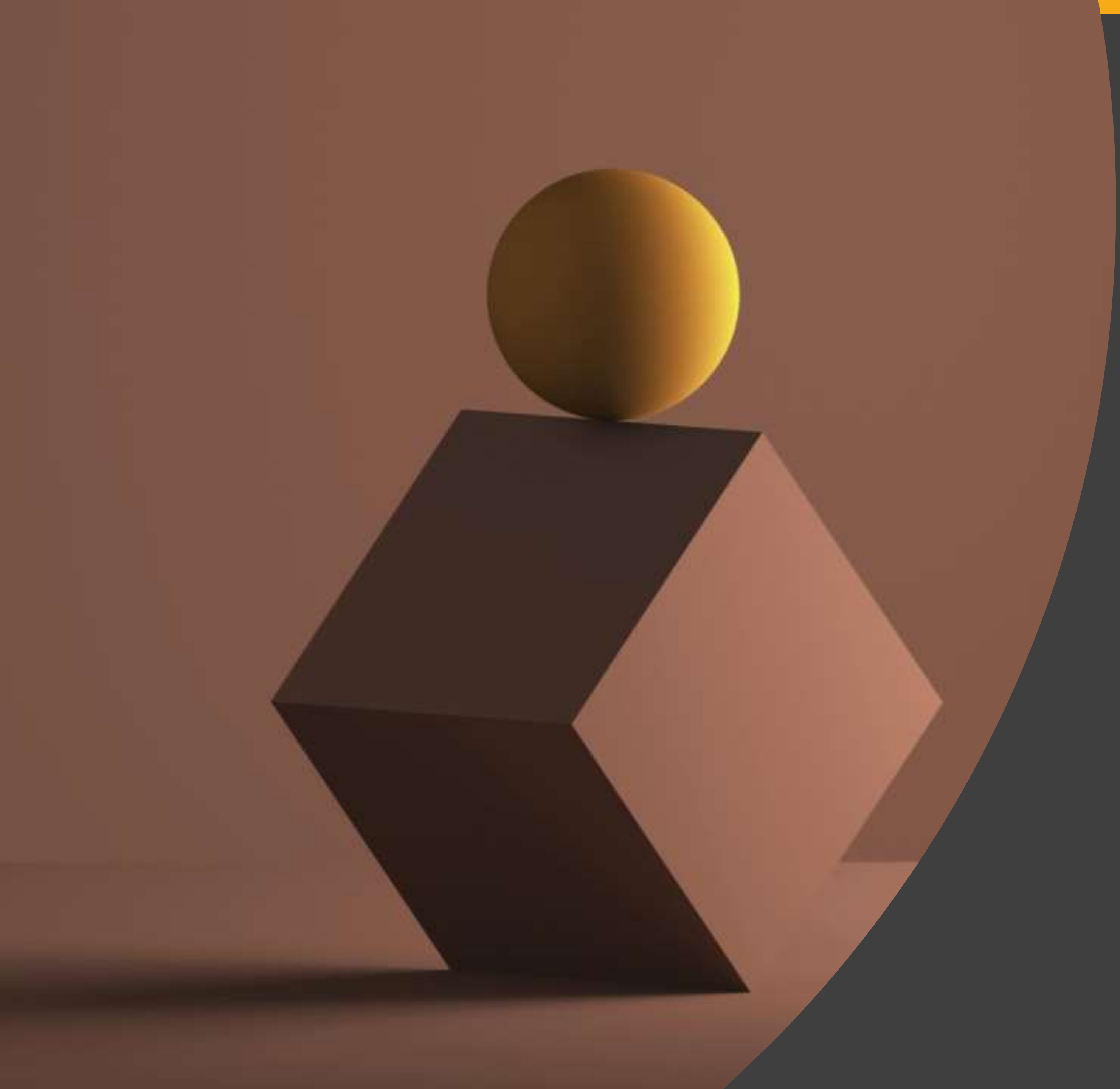
```
lea si, dat1
lea di, dat2
mov cx, 0dh
cld
REPE CMPSB
```

- Finally, How to calculate the Index where the mismatch happens?



```
    sub di, offset dat2
    mov bx, di
    dec bx
    lea si, res
    mov [si],bx

.exit
4 references
end
```



Thankyou