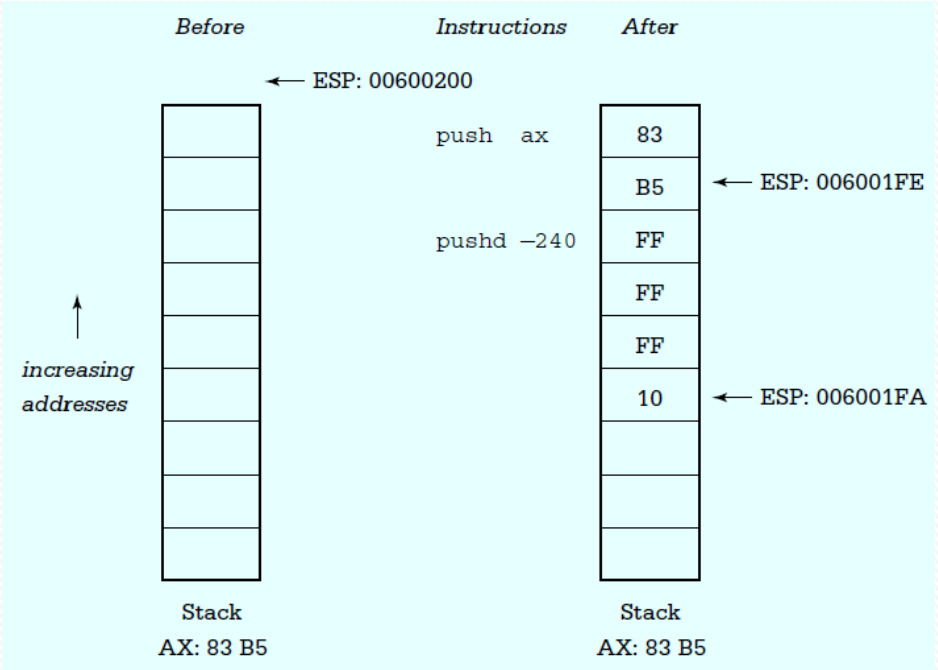
EC325 Microprocessors String Operations

Yasser F. O. Mohammad

REMINDER 1: Push instruction

- push source
  1. Decrements ESP by the size of *source*.
  2. Copies *source* to the location pointed to by ESP.



It grows downward !!!!

REMINDER 2: Procedures

- The way to implement functions and function calls in IA32
- Always comes in the code segment (after .CODE)
- Has the following anatomy:

```
label PROC [[distance]] [[langtype]] [[visibility]] [[<prologuearg>]] [[USES
            reglist]] [[, parameter [[:tag]]]]...

            statements
            [ret]
label ENDP
```

REMINDER 3: How to call a procedure

- call procedureLabel
  - Does not by itself do any parameter passing
  - You do parameter passing yourself!!!!!!
  - Does two things
    1. Pushes the return address to the stack
    2. Jumps to the address of the procedure

  - As in JMP, ±32K displacement is added to EIP/IP to do the jump

REMINDER 4: INT*

- INT *number*
  1. Calculate IV=number * 4 or 8 (Real/Protected)
  2. Push flags
  3. Clear T and I flags (Traps and hardware interrupts)
  4. Push CS
  5. Read new CS from CS:[IV]
  6. Push IP/EIP onto the stack
  7. Read new IP/EIP from CS:[IV+2]
  8. Jump to new CS:IP/EIP

Used for system calls (2 bytes) instead of FAR calls (5 bytes)

What is a string for IA32?

- An array of Bytes, Words, or Double Words

```
response      BYTE    20 DUP (?)
label1        BYTE    'The results are ', 0
wordString    WORD    50 DUP (?)
arrayD        DWORD   60 DUP (0)
```

General Info About String Instructions

- Source is always in DS:ESI
- Destination (if any) is always in ES:EDI
- To know the size of each element:
  1. Add two operands that are ignored but their size used (e.g. movs ax,bx)
  2. Add suffixes to instructor
     1. b (BYTE)
     2. w (WORD)
     3. d (DWORD)
- ESI/EDI are incremented/decremented after execution.
- Direction is controlled by DF (Direction Flag)
  - 1 means decrement (right to left)
  - 0 means increment (left to right)

Direction Control

- CLD
  - Clear Direction (Auto-increment)
- STD
  - Set Direction (Auto-decrement)

String Instructions

- MOVS[B|W|D]
  - Moves a string
- SCAS[B|W|D]
  - Scans a string
- STOS[B|W|D]
  - Stores a string
- LODS[B|W|D]
  - Loads a string
- CMPS[B|W|D]
  - Compare strings

MOVS

- MOVS[B|W|D]
  - Moves one element from DS:[ESI] to ES:[EDI]
  - IF DF==0 THEN ESI++ and EDI++
  - IF DF==1 THEN ESI-- and EDI—
- Does not affect any flags

Example

```
strcopy     PROC NEAR32

; Procedure to copy string until null byte in source is copied.
; It is assumed that destination location is long enough for copy.

; Parameters are passed on the stack:
;    (1)  address of destination
;    (2)  address of source
            push   ebp                ;save base pointer
            mov    ebp,esp            ;copy stack pointer

            push   edi                ;save registers and flags
            push   esi
            pushf

            mov    esi,[ebp+8]        ;initial source address
            mov    edi,[ebp+12]       ;destination
            cld                       ;clear direction flag
whileNoNull:
            cmp    BYTE PTR [esi],0   ;null source byte?
            je     endWhileNoNull     ;stop copying if null
            movsb                     ;copy  one byte
            jmp    whileNoNull        ;go check next byte
endWhileNoNull:
            mov    BYTE PTR [edi],0   ;terminate destination string

            popf                      ;restore flags and registers
            pop    esi
            pop    edi
            pop    ebp
            ret    8                  ;exit procedure, discarding parameters
strcopy     ENDP
```

Repeating using REP

- REP INSTRUCTION
  - E.g. REP MOVS

- While CX>0

  perform INSTRUCTION

  CX=CX- 1

  END

Other REPS

- REPZ/REPE
  - While CX>0
    perform INSTRUCTION
    CX=CX- 1
    IF ZF==1
      BREAK
    END

- REPNZ/REPNE
  - While CX>0
    perform INSTRUCTION
    CX=CX- 1
    IF ZF==0
      BREAK
    END

CMPS

- CMPS[B|W|D]
  - DS:[ESI] - ES:[EDI] (updates flags)
  - IF DF==0 THEN ESI++ and EDI++
  - IF DF==1 THEN ESI-- and EDI—

SCAS

- SCAS[B|W|D]
  - AL/AX/EAX – BYTE/WORD/DWORD PTR ES:[EDI]
    (updates flags)
  - IF DF==0 THEN EDI++
  - IF DF==1 THEN EDI—

STOS

- STOS[B|W|D]
  - MOV BYTE/WORD/DWORD PTR ES:[EDI] , AL/AX/EAX (updates flags)
  - IF DF==0 THEN EDI++
  - IF DF==1 THEN EDI—

LODS

- LODS[B|W|D]
  - MOV AL/AX/EAX , BYTE/WORD/DWORD PTR ES:[ESI]
    (updates flags)
  - IF DF==0 THEN ESI++
  - IF DF==1 THEN ESI—

XLAT

- Uses a table to translate
- *n* is converted to [EBX+*n*]
- The input is put into AL before XLAT

```
table       BYTE   48 DUP (' '), '0123456789', 7 DUP (' ')
            BYTE   'abcdefghijklmnopqrstuvwxyz', 6 DUP (' ')
            BYTE   'abcdefghijklmnopqrstuvwxyz', 133 DUP (' ')
```

```
            mov    ecx, strLength ; string length
            lea    ebx, table     ; address of translation table
            lea    esi, string    ; address of string
            lea    edi, string    ; destination also string
forIndex:   lodsb                 ; copy next character to AL
            xlat                  ; translate character
            stosb                 ; copy character back into string
            loop   forIndex       ; repeat for all characters
```