COURSE: UCS1502 - MICROPROCESSORS AND INTERFACING

Stack, procedure and macro

Dr. K. R. Sarath Chandran
Assistant Professor, Dept. of CSE

This presentation covers

- Details of stack, procedure and macro

Learning Outcome of this module

- To understand procedures, stack and macro in detail

The 8086 stack

```
assume cs:code,ds:data,ss:stack
data segment
---
data ends
stack segment
                values dw 40 dup(0)
                stacktop label word
stack ends
code segment
      start:    mov ax,data
                 mov ds,ax
                mov ax,stack
                mov ss,ax
                mov sp,offset stacktop
                ----
                ----
                ----

                mov ah,4ch
                int 21h
                code ends
end start
```

Required program additions when using a stack

- Define the stack
- Load SS with the base address of stack segment
- Load SP with the offset of the top of stack

Assume SS=7000,
Here stack area will be for 80 bytes (7000-704f).
So SP = 0050.
The physical address of the top of the stack will be 70050H.

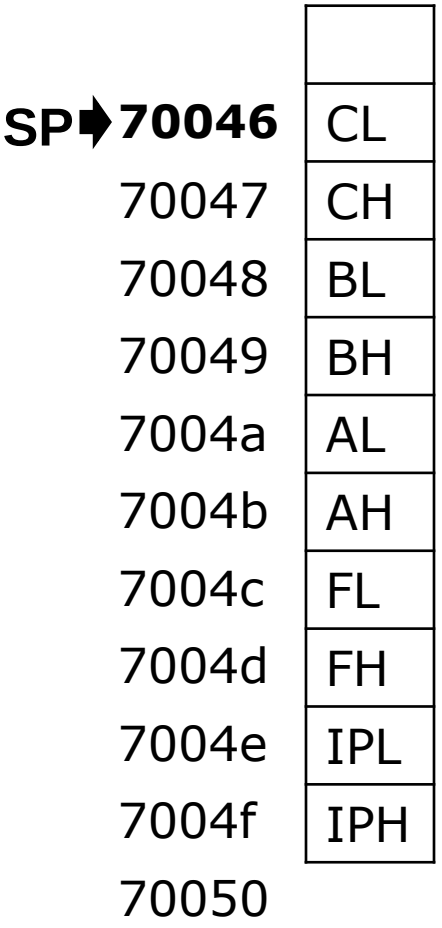
The 8086 stack and procedure

```
assume cs:code,ds:data,ss:stack
data segment
---
data ends
stack segment
            values dw 40 dup(0)
            stacktop label word
stack ends
code segment
    start:  mov ax,data
             mov ds,ax
            mov ax,stack
            mov ss,ax
            mov sp, offset stacktop
            ----
            ----
            ----

            mov ah,4ch
            int 21h
            code ends
end start
```

Initially the physical address of the top of the stack will be 70050H.

```
Sqrt proc near
        pushf
        push ax
        push bx
        push cx
        ----;body
        pop cx
        pop bx
        pop ax
        popf
        ret
Sqrt    endp
        ----

call sqrt
```

Snapshot of stack during the execution of the body of sqrt



Required program additions when using a stack

The 8086 stack and procedure

If it is a far call,

Segment 1

```
        ----
        ----
        ----

        call sqrt
        ---
        ---
```

Segment 2

```
        ----
Sqrt proc far
        pushf
        push ax
        push bx
        push cx
        ----;body
        pop cx
        pop bx
        pop ax
        popf
        ret
Sqrt    endp
        ----
```

Snapshot of stack during the execution of the body of sqrt

SP	**70044**	CL
	70045	CH
	70046	BL
	70047	BH
	70048	AL
	70049	AH
	7004a	FL
	7004b	FH
	7004c	IPL
	7004d	IPH
	7004e	CSL
	7004f	CSH
	70050	

The 8086 stack and procedure

procmain.asm

```
assume cs:code,ss:stack
stack segment
        values dw 40 dup(0)
        stacktop label word
stack ends

procedures segment public
extrn sqrt: far
procedures ends
;let assembler know that sqrt is a label of
;type FAR and is located in the segmet
;procedures
code segment
start:  mov ax,stack
        mov ss,ax
        mov sp, offset stacktop
        call sqrt
        mov ah,4ch
        int 21h
        code ends
end start
```

proc.asm

```
assume cs:procedures, ds:data
DATA SEGMENT
MESSAGE DB "THIS IS THE STRING$"
DATA ENDS
public sqrt ;make sqrt available to all modules
procedures segment public
sqrt proc far
        mov ax,data
        mov ds,ax
        MOV AH,09 ; DOS FUNCTION #9; will print all
                    characters from DX address to '$'.
        MOV DX,0000h ; OFFSET OF THE STRING
        INT 21H
        ret
 sqrt endp
procedures ends
end
```

The 8086 stack and procedure

```
D:\>link procmain.obj proc.obj;
```

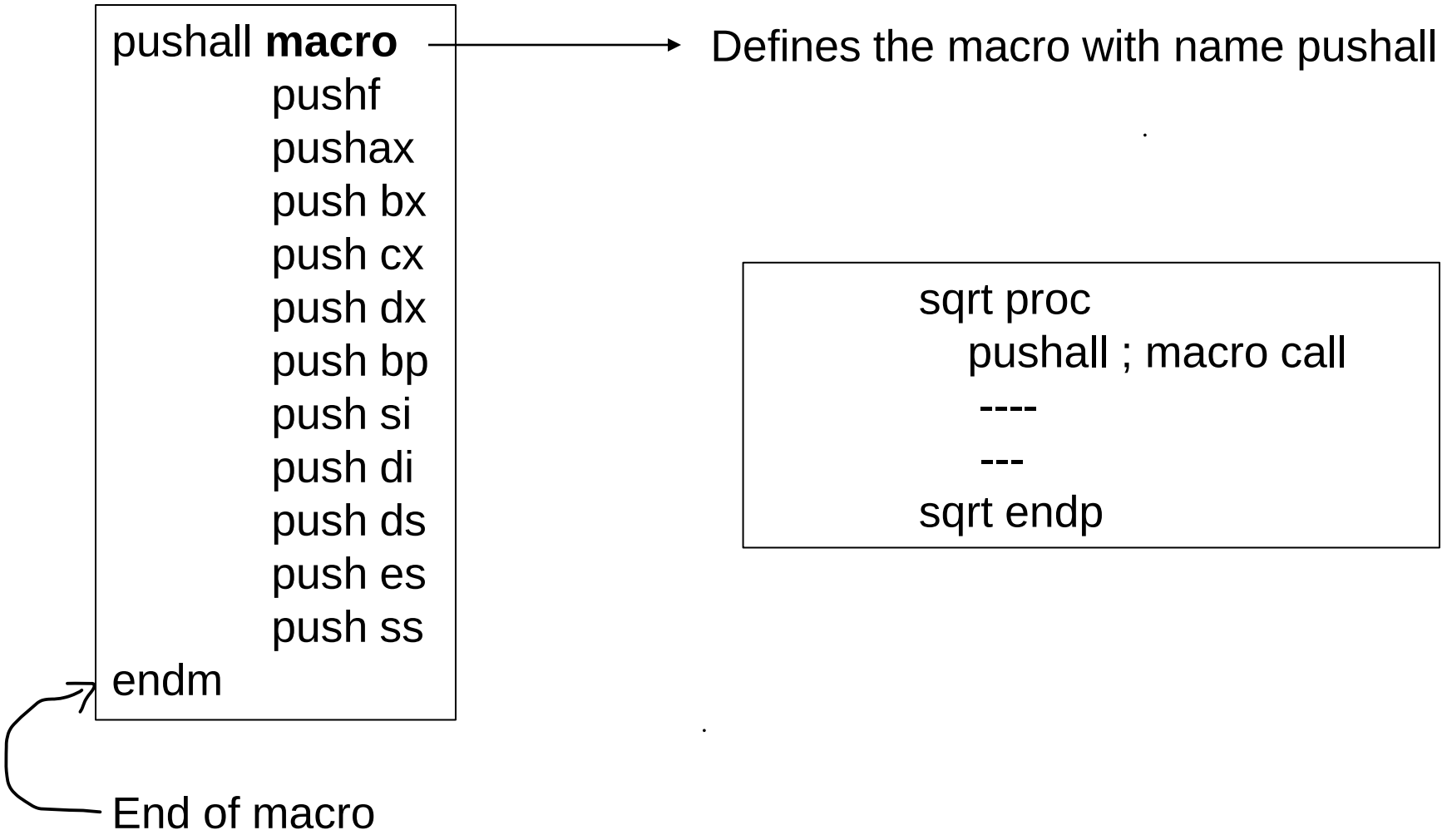
```
D:\>debug procmain.exe
-u
0770:0000 B86A07        MOV     AX,076A
0770:0003 8ED0          MOV     SS,AX
0770:0005 BC5000        MOV     SP,0050
0770:0008 9A00006F07    CALL    076F:0000
0770:000D B44C          MOV     AH,4C
0770:000F CD21          INT     21
```

```
-u 076f:0000
076F:0000 B87207        MOV     AX,0772
076F:0003 8ED8          MOV     DS,AX
076F:0005 B409          MOV     AH,09
076F:0007 BA0000        MOV     DX,0000
076F:000A CD21          INT     21
076F:000C CB            RETF
```

```
-g
THIS IS THE STRING
Program terminated normally
```

RETF (return far) will pop both the instruction pointer (IP) and the code segment (CS).

Macro


pushall **macro**
pushf
pushax
push bx
push cx
push dx
push bp
push si
push di
push ds
push es
push ss
endm
Defines the macro with name pushall
sqrt proc
pushall ; macro call

---
sqrt endp
End of macro

Passing parameters to macro

Macro definition

```
moveascii  macro number, source, destination
           mov cx, number
           LEA SI,SOURCE
           LEA DI, DESTINATION
           CLD
           REP MOVSB
           endm
```

```
Moveascii 05h, stringstart, stringdest
```

Macro call

After expansion

```
mov cx, 05h
LEA SI, stringstart
LEA DI, stringdest
CLD
REP MOVSB
```

Difference between procedure and macro?

References

- Doughlas V. Hall, “Microprocessors and Interfacing, Programming and Hardware”, Second Edition, TMH.

Thank you