Simple method to solve mathematical equation by using procedure in 8086

RavindraSundarlalKakade[#1], VasimrajSiraj Tamboli[*2],AvinashBalasahebAnap[*3]

[#] Senior Lecturer in Computer Depatment, Pad.Dr.VitthalraoVilkhepatil Polytechnic Loni

Abstract

The repeated group of instructions in a large program can be written separately from the main program. This sub program is called as procedure in ALP.

Definition of Procedure:-

- Procedure is a set of statements that can be processed independently from the main program. For defining procedure PROC & ENDP assembler directives are used.
- The PROC indicates the beginning of procedure and ENDP directive indicate end of procedure.
- The procedure must be defined in code segment

Difference between Near Procedure and For Procedure:-

NEAR Procedure	**FAR Procedure:**
1) A Procedure is written in the same code segment called as NEAR Procedure.	1) A Procedure is written in the different code segment called as FAR Procedure.
2) The content of CS is not stored	2) The content of CS is also stored with offset.
3) In NEAR CALL the content of SP is decrement by 2 and the content of offset address IP is stored.	3) In FAR CALL the content of SP is decrement by 2 and values of CS is loaded. Then SP is again decrement by 2 and IP is loaded.
4) Syntax Procedure_name PROC NEAR - - - - - - - - - - - - - - - - (statements) - - - - - - - - - - - - - - - - RET Procedure_name ENDP	4) Syntax Procedure_name PROC FAR - - - - - - - - - - - - - - - - (statements) - - - - - - - - - - - - - - - - RET Procedure_name ENDP
5) e.g. Addition PROC NEAR	5) e.g. Addition PROC FAR

NEAR and FAR Procedures:

NEAR Procedure:
- A Procedure is written in the same code segment called as NEAR Procedure.
- Only instruction Pointer (IP) register contents will be changed in near procedure.

FAR Procedure:
- A Procedure is written in the different code segment called as FAR Procedure.

- In FAR procedure both instruction Pointer (IP) and Code segment (CS) register contents will be changed.

General structure of Procedure:

```
Procedure_name  PROC NEAR / FAR
-   -   -   -   -   -   -  -
-   -   -   -   -   -   -  -
(statements)
-   -   -   -   -   -   -  -
-   -   -   -   -   -   -  -
RET
Procedure_name  ENDP
```

CALL & RET instructions:

CALL Instruction: CALL a procedure

- CALL instruction used to transfer program execution to a procedure.
- CALL instruction makes two operations

- 1) When CALL executes first it stores the address of the instruction after CALL instruction on Stack. This address is called as Return Address.
- 2) Second operation of CALL instruction is to change contents of the IP register.

Two basic types of CALL

1) NEAR CALL - It is a call to a procedure in the same code segment.
   i.e. intra - segment CALL
2) FAR CALL - It is a call to a procedure I n the different code segment.
   i.e. inter - segment CALL

Operation for NEAR CALL:

Format- NEAR CALL PROC

* If NEAR CALL PROC - then

1) SP ← SP - 2
   save IP on stack
   IP ← address of procedure

* If FAR CALL PROC - then

1) SP ← SP - 2 (SP ← CS i.e. save CS on stack)
2) CS ← new segment base address of the called procedure.
3) SP ← SP - 2 (SP ← IP i.e. save IP on stack)
4) IP ← new offset address of the called procedure.

For example:

1) CALL Addition: Direct within the same code segment that calls the procedure of name addition.

2) CALL BX: Indirect within the segment, where BX contains the offset of the first instruction of the procedure and replace the content of IP with content of BX with register.

How the procedure is called from main program?

- The repeated group of instruction in a large program can be written separately from the main program. This subprogram is called a Procedure.

- Procedure can be written in same code segment or in different code segment.
- Procedure is called from main program by using CALL instruction.
- Two types of CALL instruction
- FAR CALL used for Inter segment procedure.
- NEAR CALL is used for Intra segment.

Format:-

```
-------------
-------------
(Program statements)
-------------
CALL Procedure_name
-------------
-------------
```

Advantages and Disadvantages of using procedure.

Advantages of procedure-

1) Large Program can be split into smaller modules.
2) Procedure reduces the size of program.
3) Debugging of errors in program & procedure can be perform easily.
4) By using procedure program development becomes easier.
5) Reuse of procedure many times is same program.
6) By using procedures it reduces work and development time.

Disadvantages of procedure-

1) CALL and RET instructions are always required to integrate with procedures.
2) Requires extra time to Link the procedure & return from it. So execution time is more.
3) For small group of instructions linking and returning back time is more than the small group of instructions procedures can not be performed.

RET Instruction:

RET - Return from Procedure

This instruction will return execution from a procedure to the next instruction after the

CALL instruction which was used to call a procedure i.e. main program.

- i.e. RET instruction transfer the control from procedure to main program.
- Stack pointer will increment by 2.
- Return address will be popped from the stack to IP.
- At the end of every CALL procedure the RET instruction must be executed.

1) NEAR RET :

```
operation-    IP  ←  content from top of stack.
              SP  ←  SP + 2
```

2) FAR RET :

```
operation-    IP  ←  content from top of stack.
              SP  ←  SP + 2
              CS  ←  content of top of stack.
              SP  ←  SP + 2
```

IRET Instruction:

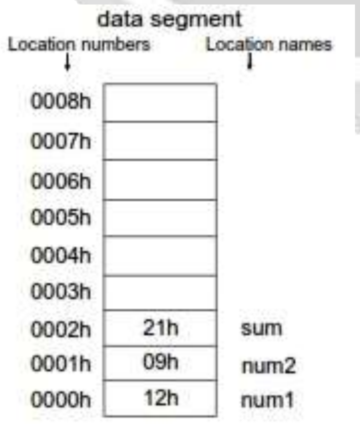
IRET - This instruction used at the end of the interrupt service procedure to return the execution to the interrupted program.

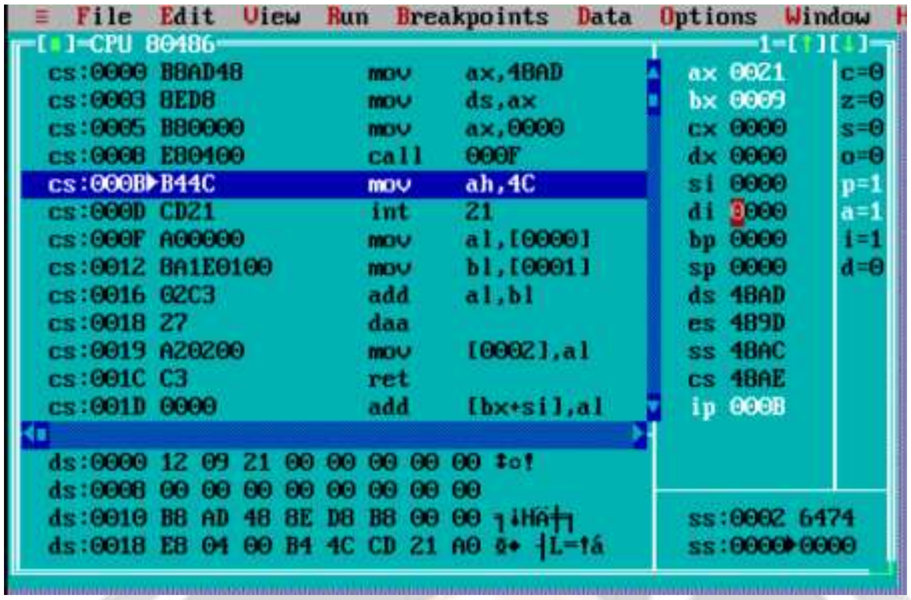
- During the execution of this instruction of IRET instruction 8086 copies the saved values of IP from the stack to IP, and the saved values of CS from stack to CS. Also saved values of Flags back to the flag register.

```
operation-      SP  ←  SP + 2   IP is popped from stack
                SP  ←  SP + 2   CS is popped from stack
                SP  ←  SP + 2   Flag register is popped from stack
```

Addition of two BCD numbers using PROCEDURE”

```
data segment
        num1 db 12h         ; first number
        num2 db 09h         ; second number
        sum db ?            ; sum variable for result
data ends

code segment                ; start of code segment
assume cs:code, ds:data
start:  mov ax,data         ;
        mov ds,ax           ; initialization of data segment
        mov ax,0000h        ; clear content of ax register

        CALL bcd_add        ; call Procedure which name is bcd_add
        mov ah,4ch
         int 21h            ; program termination

        bcd_add PROC        ; start of Procedure bcd_add
        mov al,num1
        mov bl,num2
        add al,bl
        DAA                 ; decimal adjustment after addition.
        mov sum,al
        RET                 ; return from Procedure bcd_add
        bcd_add ENDP        ; end of Procedure bcd_add

         code ends          ; end of code segment
         end start          ; end of program
```



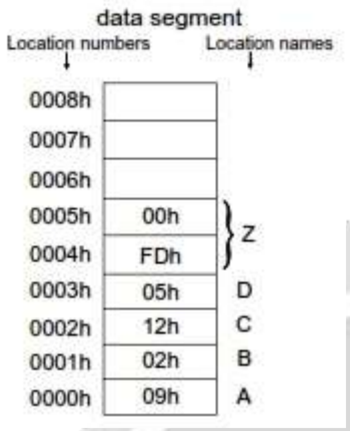


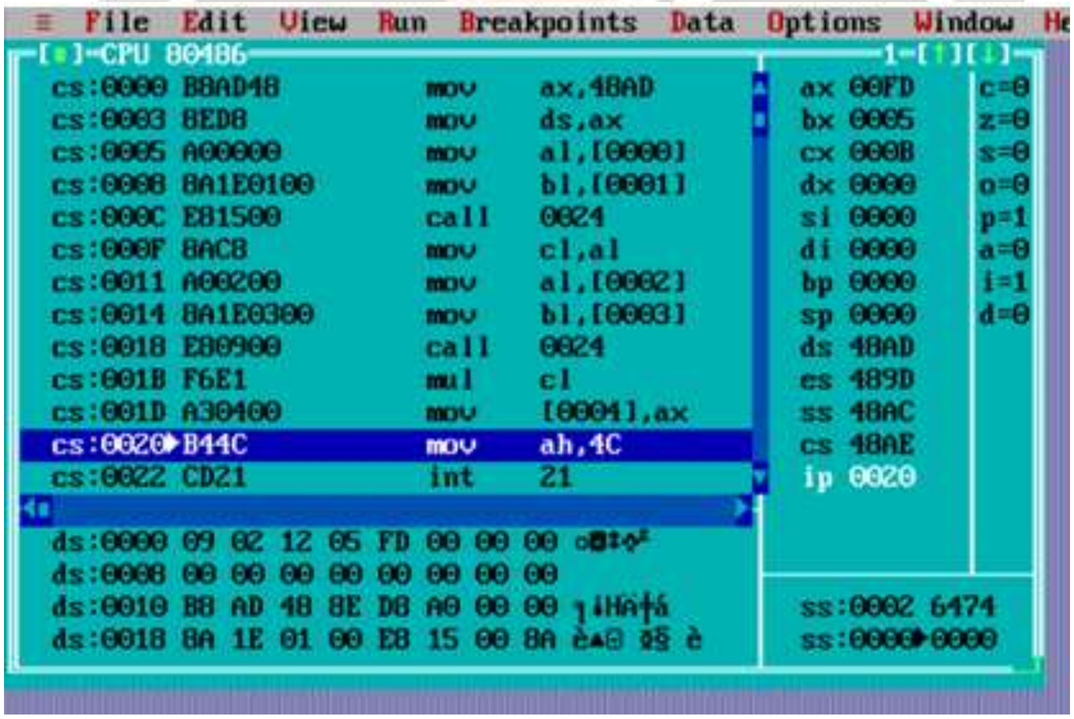
```
cs:0000 B8AD48       mov   ax,48AD          ax 0021   c=0
cs:0003 8ED8         mov   ds,ax            bx 0009   z=0
cs:0005 B80000       mov   ax,0000          cx 0000   s=0
cs:0008 E80400       call  000F             dx 0000   o=0
cs:000B►B44C         mov   ah,4C            si 0000   p=1
cs:000D CD21         int   21               di 0000   a=1
cs:000F A00000       mov   al,[0000]        bp 0000   i=1
cs:0012 8A1E0100     mov   bl,[0001]        sp 0000   d=0
cs:0016 02C3         add   al,bl            ds 48AD
cs:0018 27           daa                    es 489D
cs:0019 A20200       mov   [0002],al        ss 48AC
cs:001C C3           ret                    cs 48AE
cs:001D 0000         add   [bx+si],al       ip 000B

ds:0000 12 09 21 00 00 00 00 00 ‡○!
ds:0008 00 00 00 00 00 00 00 00
ds:0010 B8 AD 48 8E D8 B8 00 00 ╕¡HÄ╪╕          ss:0002 6474
ds:0018 E8 04 00 B4 4C CD 21 A0 Φ♦ ┤L═!á       ss:0000►0000
```

ALP using procedure to solve equation such as Z= (A+B)*(C+D)

```
data segment
A db 09H
B db 02H
C db 12H
D db 05H
Z dw ?
data ends

code segment
assume cs:code, ds:data
start:      mov ax,data
            mov ds,ax
            mov al,A
            mov bl,B

            CALL operation PROC
            mov cl,al
            mov al,C
            mov bl,D

            CALL operation PROC

            mul cl
            mov Z,ax
            mov ah,4ch
            int 21h

            operation PROC
            add al,bl
            RET
            Operation ENDP
```

code ends
end start





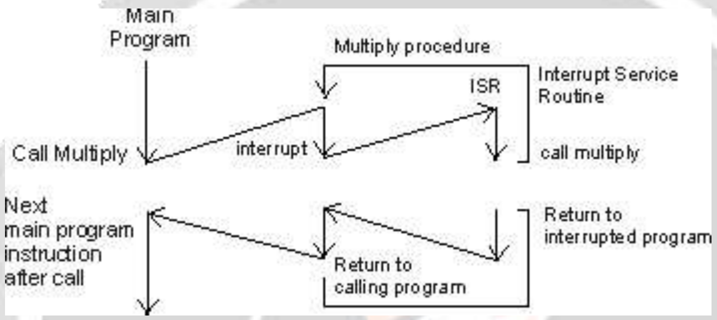
```
[■]-CPU 80486                                          1-[↑][↓]
cs:0000 B8AD48        mov   ax,48AD        ax 00FD    c=0
cs:0003 8ED8          mov   ds,ax          bx 0005    z=0
cs:0005 A00000        mov   al,[0000]      cx 000B    s=0
cs:0008 8A1E0100      mov   bl,[0001]      dx 0000    o=0
cs:000C E81500        call  0024           si 0000    p=1
cs:000F 8AC8          mov   cl,al          di 0000    a=0
cs:0011 A00200        mov   al,[0002]      bp 0000    i=1
cs:0014 8A1E0300      mov   bl,[0003]      sp 0000    d=0
cs:0018 E80900        call  0024           ds 48AD
cs:001B F6E1          mul   cl             es 489D
cs:001D A30400        mov   [0004],ax      ss 48AC
cs:0020►B44C          mov   ah,4C          cs 48AE
cs:0022 CD21          int   21             ip 0020

ds:0000 09 02 12 05 FD 00 00 00 ○☻↕♣²
ds:0008 00 00 00 00 00 00 00 00
ds:0010 B8 AD 48 8E D8 A0 00 00 ╕¡HÄ╪á       ss:0002 6474
ds:0018 8A 1E 01 00 E8 15 00 8A è▲☺ Φ§ è      ss:0000►0000
```

Re-enterent Procedure:-
A procedure is said to be reentrant, if it can be interrupted, use and re-entered without losing or overwriting over anything.

- To be re-enterant, procedure must first push all flags and registers used in the procedure. It should also use only registers or stack to pass parameters.

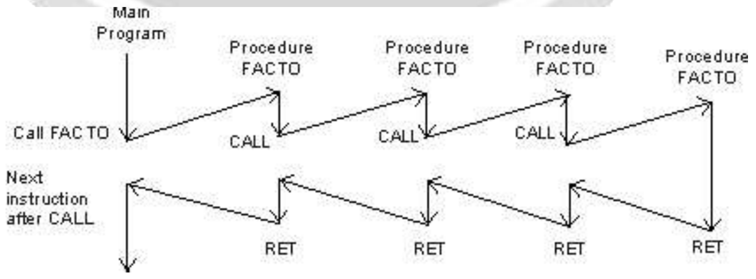
- In some situation it may happen that the procedure 1 is called from main program, procedure 2 is called from procedure 1 is again called from called from procedure 2. In this situation program execution flow returns in the procedure 1. These types of procedures are called s re-enterant procedures.

- The factorial (multiply) procedure must be written in such a way that it can be interrupted used re-enteredwithout losing or overwriting anything is called Re-enterent procedure.


Main Program
Multiply procedure
ISR
Interrupt Service Routine
Call Multiply
interrupt
call multiply
Next main program instruction after call
Return to calling program
Return to interrupted program

Recursive Procedure
- It is a procedure which call itself.
- Recursive procedures are used to work with complex data structure like trees.
- If procedure is called with N (recursive depth) then N is decremented by one after each

procedure CALL and the procedure is called until n=0.
- Recursive procedure takes less time to implement a particular task. But it needs certain condition for it's termination.



Conclusion:-

This report presents an introduction to Procedure used in 8086 Microprocessor Programming. Two methods can be used near procedure and far procedure. This was an introduction to the main aim of paper is that how to write and procedure and how to call in main program. The figure of data segment is innovatively used to show the contents used in program and what is the result obtained.

.

References :-

[1] "Microprocessor & interfacing (programming & hardware)" by Douglas V-Hall.
[2] "The 8088 and 8086 Microprocessors" by Walter A. Triebel, Avtar Singh
[3] "Microprocessor & Controllers" by Latha, C., Murugeshwari, B