Programming with Assembler

- Using the Turbo Assembler

- I recommend that you get Sivarama P. Dam-damudi’s book. It is available at Amazon
Structure of Assembly Programs

TITLE brief title of program
COMMENT |

Objectives:
Inputs:
Outputs:

| .MODEL SMALL |

.STACK 100H ; defines a 256 byte stack
.DATA
(data goes here)

.CODE
.486

INCLUDE io.mac ; include io routines
main PROC

.STARTUP ; set up segments
.
.
(code goes here)
.
.
.EXIT ; returns control

main ENDP
END main
Assembling and Linking

1. *TASM sample*
   generates *sample.obj* from *sample.asm*

2. You will also need to assemble *io.mac*.

3. To make the executable use the linker
   *TLINK sample io*
   this generates *sample.exe*
Procedures and the stack

The Pentium Stack

- Only words or double words are saved on the stack.

- The stack grows towards lower memory adresses

- SS:SP, always point to the lower byte of the last word on the stack.
Stack Operations

- To make a new stack, (you ALWAYS need one)

  .STACK 100H ; or however big you want it

- push source
  pop destination

  pushf ; push 16-bit flags
  popf ; pop 16-bit flags
Uses of the stack

1. As a scratch pad

   push   val1
   push   val2
   pop    val1
   pop    val2

2. For transfer of control.

3. For parameter passing
Assembler Directives for procedures

- proc-name PROC NEAR
- proc-name PROC FAR
- proc-name ENDP
- call proc-name
- ret
Control transfer

main PROC

.....
call sum
mov AX,BX

.....
main ENDP

sum PROC

push BP

.....

sum ENDP
Parameter Passing Using registers

main PROC
.STARTUP
PutStr prompt_msg1 ; request first number
GetInt CX ; CX := first number

GetInt DX ; DX := second number

PutStr prompt_msg2 ; request second number

PutStr sum_msg ; display sum
PutInt AX

done:
.EXIT
main ENDP
; Procedure sum receives two integers in CX and DX.
; The sum of the two integers is returned in AX.
;-------------------------------------------------------------------

sum PROC
    mov AX,CX ; sum := first number
    add AX,DX ; sum := sum + second number
    ret
sum ENDP

END main
Parameter Passing Using the stack

main PROC
  .STARTUP
  PutStr prompt_msg1 ; request first number
 GetInt CX ; CX := first number
  nuln
  PutStr prompt_msg2 ; request second number
 GetInt DX ; DX := second number
  nuln
  push CX ; place first number on stack
  push DX ; place second number on stack
call sum ; returns sum in AX
  PutStr sum_msg ; display sum
  PutInt AX
  nuln
done:
  .EXIT
main ENDP
; Procedure sum receives two integers via the stack.
; The sum of the two integers is returned in AX.

sum PROC
    push BP
    ; we will use BP, so save it for later use
    mov  BP,SP
    mov  AX,[BP+6]
    ; sum := first number
    add  AX,[BP+4]
    ; sum := sum + second number
    pop  BP
    ; restore BP
    ret  4
    ; return and clear parameters
sum ENDP
END   main
A Variable Number of parameters

main PROC
    .STARTUP
    PutStr prompt_msg ; request input numbers:
    nwln
    sub CX,CX ; CX keeps number count
    read_number:
        GetInt AX ; read input number
        nwln
        cmp AX,0 ; if the number is zero
        je stop_reading ; no more numbers to read
        push AX ; place the number on stack
        inc CX ; increment number count
        jmp read_number
stop_reading:
push CX ; place number count on
call variable_sum ; returns sum in AX
; clear parameter space on the stack
inc CX ; increment CX to include
add CX,CX ; CX := CX * 2 (space for
add SP,CX ; update SP to clear parameter
; space on the stack
PutStr sum_msg ; display the sum
PutInt AX

ln

done:

.EXIT

main ENDP
This procedure receives variable number of integers in stack. The last parameter pushed on the stack shows the number of integers to be added. Sum is returned.

variable_sum PROC
  push BP ; save BP - procedure using BP
  mov BP,SP ; copy SP to BP
  push BX ; save BX and CX
  push CX

  mov CX,[BP+4] ; CX := # of integers to add
  mov BX,BP
  add BX,6 ; BX := pointer to first integer
  sub AX,AX ; sum := 0
add_loop:
  add   AX, SS:[BX] ; sum := sum + next num
  add   BX, 2 ; BX points to the next
  loop  add_loop ; repeat count in CX
  pop   CX ; restore registers
  pop   BX
  pop   BP
  ret ; parameter space clean

variable_sum ENDP
END main