# The Stack and Introduction to Procedures

# Outline

- The stack
- A stack application
- Terminology of procedures
- CALL and  RET
- An example of a procedure

# The Stack

- A stack is one-dimensional data structure.
- Items are added and removed from one end of the structure that is, it is processed in a “last-in, first-out” manner.
- The most recent addition to the stack is called the top of the stack.
- A program must set aside a block of memory to hold the stack. We have been doing this by declaring a stack segment; for example,

```
.STACK    100H
```

# The Stack

- When the program is assembled and loaded in memory, SS will contain the segment number of the stack segment.
- For the preceding stack declaration, SP, the stack pointer, is initialized to 100h. This represent the empty stack position. When the stack is non empty, SP contains the offset address of the top of the stack.

# PUSH and PUSHF Instruction

- To add a new word to the stack we PUSH it on.
- The syntax is

  PUSH source ;where source is a 16-bit register or memory word.

- For example : PUSH AX
- Execution of PUSH cause the following to happen:
  1. SP is decreased by 2.
  2. A copy of the source content is moved to the address specified by SS:SP. The source is unchanged.

# PUSH and PUSHF Instruction

- The instruction PUSHF, which has no operands, pushes the contents of the FLAGS register onto the stack.
- Initially, SP contains the offset address of the memory location immediately following the stack segment; the first PUSH decreases SP by 2, making it point to the last word in the stack segment.
- Because each PUSH decreases SP, the stack grows toward the beginning of memory.

# POP and POPF Instruction

- To remove the top item from the stack we POP it.
- The syntax is

  POP destination ;where destination is a 16-bit register (except IP) or memory word.

- For example : POP BX
- Executing POP causes this to happen:
  1. The content of SS:SP (the top of the stack) is moved to the destination.
  2. SP is increased by 2.
- The instruction POPF pops the top of the stack into the FLAGS register.

# Stack Instructions

- There is no effect of PUSH, PUSHF, POP, POPF on the flags.
- Note that PUSH and POP are word operations, so a byte instruction such as PUSH DL is illegal.
- A push of immediate data, such as PUSH 2 is also illegal.
- In addition to the user's program, the operating system uses the stack for its own purposes.
- For example, to implement the INT 21h functions, DOS saves any registers it uses on the stack and restores them when the interrupt routine is completed. This does not cause a problem for the user because any values DOS pushes onto the stack are popped off by DOS before it returns control to the user's program.

# Procedure Declaration

- Syntax:

```
name PROC type
    body of the procedure
    RET
name ENDP
```

- Name is the user-defined name of the procedure.
- The optional operand type is NEAR or FAR (if type is omitted, NEAR is assumed).
- NEAR means that the statement that calls the procedure is in the same segment as the procedure itself. FAR means that the calling statement is in a different segment.
- In the following, we assume all procedures are NEAR.

# CALL Instruction

- To invoke a procedure, the CALL instruction is used.
- There are two kinds of procedure calls, direct and indirect.
- Syntax:

  direct procedure: CALL name; where name is the name of a procedure.

  indirect procedure: CALL address_expression; where

  address_expression specifics a register or memory location containing the address of a procedure.

# CALL Instruction

- Executing a CALL instruction causes the following to happen:
  1. The return address to the calling program is saved on the stack. This is the offset of the next instruction after the CALL statement. The segment: offset of this instruction is in CS: IP at the time the call is executed.
  2. IP gets the offset address of the first instruction of the procedure. This transfers control to the procedure.

# RET Instruction

- The RET (return) instruction causes control to transfer back to the calling procedure.
- Syntax:

```
RET pop_value  ;integer argument pop_value is optional.
```

- For a NEAR procedure execution of RET causes the stack to be popped into IP.
- If a pop_value N is specified, it is added to SP, and thus has the effect of removing N additional bytes from the stack.
- CS:IP now contains the segment :offset of the return address, and control returns to the calling program.
- Every procedure (except the main procedure) should have a RET someplace
- Usually it's the last statement in the procedure