Lesson 4. Stack, Procedures and Macros

Computer Structure and Organization
Graduated in Computer Sciences / Graduated in Computer Engineering
Lesson 4: Stack, Procedures and Macros

Contents

- Bit instructions
- The stack
- How to define procedures
- Type of procedures: NEAR y FAR
- How to pass parameters to a procedure:
  - Registers
  - The Stack
- Structures and parameters passing
- Macros
- Procedures vs. Macros
Lesson 4: Stack, Procedures and Macros

Bit instructions (I)

- **Mnemonic:** TEST

- **Format:** TEST target, source

- **Description:**
  This function is a logical AND over source and target. The difference is that the result is not stored in anywhere. This instruction only modifies flags register bits.

- **Example:**
  \[
  \begin{align*}
  \text{; AX} &= 1234, \text{ BX} = 0000 \\
  \text{TEST AX, BX} &; \text{ AX} = 1234, \text{ BX} = 0000, \text{ Result} = 0000 \\
  \text{JZ Es_Cero} &; \text{ IF Zero jumps to Es_Cero label}
  \end{align*}
  \]
Lesson 4: Stack, Procedures and Macros

Bit instructions (II)

- **Mnemonic:** CLI
- **Format:** CLI
- **Description:**
  Disable masked interrupts handling.
  Unmasked interrupt can not be disabled
- **Example:**
  - CLI
Lesson 4: Stack, Procedures and Macros

Bit instructions (III)

- **Mnemonic:** STI

- **Format:** STI

- **Description:** Enable interrupts handling

- **Example:**
  - STI
Lesson 4: Stack, Procedures and Macros

Bit instructions (IV)

- **Mnemonic:** CLC

- **Format:** CLC

- **Description:**
  Carry flag is set to zero

- **Example:**
  - CLC
Lesson 4: Stack, Procedures and Macros

Bit instructions (V)

- **Mnemonic**: STC
- **Format**: STC
- **Description**: Carry flag is set to one
- **Example**: STC
Bit instructions (VI)

- **Mnemonic:** CLD

- **Format:** CLD

- **Description:**
  Direction flag is set to zero (used by string instructions)

- **Example:**
  - CLD
Lesson 4: Stack, Procedures and Macros

Bit instructions (and VII)

- **Mnemonic:** STD
- **Format:** STD
- **Description:**
  Direction flag is set to one (used by string instructions)
- **Example:**
  - STD
Lesson 4: Stack, Procedures and Macros

The Stack

- Portion of memory that handles return addresses of procedures and interrupt services
- LIFO structure (Last In First Out)
- SS and SP are dedicate registers to point to the top of the stack
- Each stack position is 16 bits length
- The stack increases to lower memory positions addresses with PUSH
- The stack decreases to upper memory positions addresses with POP
Definición de procedimientos

- Procedimientos agrupan instrucciones y permiten llamarlas desde diferentes partes del programa sin necesidad de repetirlas.

- Para llamar a un procedimiento: CALL

- Última instrucción de un procedimiento: RET

- Dirección de retorno del procedimiento se almacena en la pila. Si el tipo del procedimiento es FAR, el segmento de dirección al que pertenece el procedimiento se almacena en la pila.

- Para definir un procedimiento se utilizan las siguientes directivas: PROC (comienzo del procedimiento) y ENDP (fin del procedimiento).
Types of procedures

- Procedures have the following attributes: **FAR** (if it will be called from a different segment it belongs to) or **NEAR** (call is within the same segment)
- Main Procedure is always FAR (implicít)

**Ejemplo de definición de un procedimiento**

```
PrintString    PROC   FAR   ; this procedure calls to 9h DOS
    MOV AH, 09h   ; services to display a character string on the
    INT 21h       ; string whose address is stored on DS:DX
    RET           ; return of the procedure
PrintString    ENDP
```
Lesson 4: Stack, Procedures and Macros

How to pass parameters to a procedure by using registers

- General purpose registers are used to pass and to receive parameters and results from a procedure
- We so few registers the i8086 has, it’s needed to save used registers by the procedure before calling it
- First set of instructions in a procedure must store on the stack all the registers that will be used in it.
- Before returning from procedure save registers must be restored. *All of them except those in which results will be returned*
How to pass parameters to a NEAR PROCEDURE. The stack (I)

Actions performed by CALLER procedure
- Parameters offset or variables themselves are in the same segment so it will be stored on the stack

**Caller Procedure:**
- MOV AX, VARIABLE1
- PUSH AX
- LEA AX, VARIABLE2
- PUSH AX
- ...
- CALL Nombre_Proc

**Stack Structure**
- SS:SP
- IP (returning address)
- Offset VARIABLEn
- Offset Variable2
- Offset Variable1
Lesson 4: Stack, Procedures and Macros

How to pass parameters to a NEAR PROCEDURE. The stack (II)

Actions performed by CALLED procedure
1. BP register is stored on the stack (PUSH BP) BP will be used to access parameters.
2. BP must point to SP (MOV BP, SP)
3. Parameters are recovered from the stack by using BP. The i parameter offset will be: \([BP] + 4 + 2 \cdot (n - i)\)

CALLER procedure:
Name_Proc PROC
PUSH BP
MOV BP,SP
PUSH [used registers]
MOV AX,[BP+4] ;VARIABLEn
...
MOV BX,[BP+4+2 \cdot (n-i)]

Estructura de la pila

BP register
IP (returning address)
Offset VARIABLEn

Offset Variable2
Offset Variable1
Lesson 4: Stack, Procedures and Macros

How to pass parameters to a NEAR PROCEDURE. The stack (and III)

Actions performed by CALLED procedure

- RET parameter indicates the stack positions to be removed when returning from CALLED procedure.
- This number will be 2xnumber of passed parameters.
- E.g. RET 8 if four parameters have been passed

**CALLED procedure:**

- POP [used registers]
- POP BP
- RET 2\cdot n
- Name_Proc ENDP

**Stack structure:**

- BP register
- IP (returning address)
- Offset VARIABLEn
- Offset Variable2
- Offset Variable1
Lesson 4: Stack, Procedures and Macros

How to pass parameters to a FAR PROCEDURE. The stack (I)

Actions to be performed by CALLER procedure

- Variables offset and segment must be stored on the stack because CALLER and CALLED procedures belongs to different memory segmentes

**CALLER procedure:**

LEA AX, VARIABLE1
PUSH DS
PUSH AX
LEA AX, VARIABLE2
PUSH DS
PUSH AX
...

CALL PROCEDIMIENTO
Lesson 4: Stack, Procedures and Macros

How to pass parameters to a FAR PROCEDURE. The stack (and II)

Actions performed by CALLED procedure

• BP register is stored on the stack (PUSH BP) BP will be used to access parameters.
• BP must point to SP (MOV BP, SP)
• Parameters are recovered from the stack by using BP. The i parameter offset will be: \([\text{BP}] + 4 + 2 \cdot (n - i)\) and the segment whose belongs to: \([\text{BP}] + 8 + 4 \cdot (n - i)\)

RET parameter of the CALLED procedure:

• RET parameter indicates the stack positions to be removed when returning from CALLED procedure.
• This number will be 4xnumber of passed parameters.
• E.g. RET 16 if four parameters have been passed
Structures and parameters passing (I)

Structures

- Structures will be defined with following directives: STRUC (start of structure) and ENDS (ends of structure)
- Group different type and size data with a single name
- StructureName.Field will be used to access to structure records
- RET argument and parameter passing can be easy by using structures

```plaintext
DNI STRUC
    Name_and_Surname   DB 50 DUP ('0')
    Address            DB 60 DUP ('0')
    Age                DB 0
    Profession         DB 15 DUP ('0')
DNI ENDS
```
Structs and parameters passing (II)

Structure code and records for parameters passing to a NEAR procedure:

Estructura STRUC

bp0    DW ? ; 2 bytes for storing BP
return DW ? ; It’s equivalent to a +2 bytes offset from the beginning of the
          ; structure (first DW named bp0)
p2     DW ? ; It’s equivalent to a +4 bytes offset from the beginning of the
          ; structure (DW + return)
P1     DW ? ; It’s equivalent to a +6 bytes offset from the beginning of the
          ; structure (DW + return + p2)

Estructura ENDS

NRET EQU OFFSET p1 - OFFSET return
Lesson 4: Stack, Procedures and Macros

Structs and parameters passing (and III)

Actions performed by NEAR CALLED procedure
1. BP register is stored on the stack (PUSH BP) BP will be used to access parameters.
2. BP must point to SP (MOV BP, SP)
3. Accessing to parameters with BP and the structure
4. RET argument will be NRET

Example of previous defined structure

```
Estructura struc
    bp0        DW ?
    retorno    DW ?
    p2         DW ?
    p1         DW ?
Estructura ends
NRET EQU OFFSET p1 - OFFSET retorno
```

CALLED procedure

```
Prueba PROC NEAR
    PUSH BP
    MOV BP, SP
    PUSH AX
    MOV AX, [BP].p1
    ADD [BP].p2, AX
    POP AX
    POP BP
    RET NRET
Prueba ENDP
```
MACROS

- MACROS are names given to some pieces of code
- Code expansion instead of jumping is done
- Extra parameters will be ignored if passed. So have instructions that likely use them
- Mind the labels inside a macro
- Labels must be local inside the macro code
Las macros (II)

- **Mnemonic:** MACRO
- **Format:** macro_name MACRO parameter_list
- **Description:** name and parameters to be passed to the macro are defined. Parameter names are separated by commas. The end of a macro definition is specified using ENDM directive without writing macro name

- **Nombre y Format:** ENDM
- **Description:** to establish the end of a macro definition
- **Example:**
  
  Add_three MACRO operand1, operand2, result
  ;macro body
  ENDM
Las macros (III)

- **Mnemonic:** LOCAL
- **Format:** LOCAL label,[label,...]
- **Description:** indicate label names to be changed when expanding a macro. This method solves duplicate label names. Must be used after macro definition line and only use it on macros.

- **Example:**
  
  ```assembly
  Delay MACRO number
  local go_on
  mov cx, number
  go_on:    loop go_on
  ENDM
  ```
### Procedures vs. Macros

<table>
<thead>
<tr>
<th>Procedures</th>
<th>Macros</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related code is unique</td>
<td>Code is repeated each time the macro is expanded</td>
</tr>
<tr>
<td>Slower than macros</td>
<td>Quicker than macros</td>
</tr>
<tr>
<td>Return to next instruction to the caller one</td>
<td>Next instruction to the macro is executed when macro ends</td>
</tr>
<tr>
<td>No much flexibility for parameters passing</td>
<td>A great flexibility in parameters passing</td>
</tr>
</tbody>
</table>