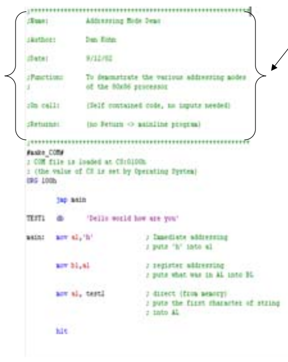
Assembly Language Programming

By Dan Kohn
University of Southern Mississippi
Computer Engineering Technology



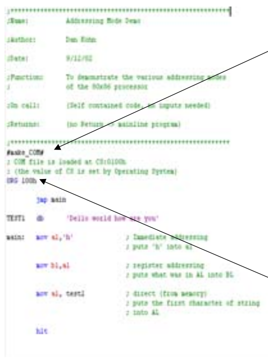
```
;**********************************************************
;Name:        Addressing Mode Demo

;Author:      Dan Kohn

;Date:        9/12/02

;Function:    To demonstrate the various addressing modes
;             of the 80x86 processor

;On call:     (Self contained code, no inputs needed)

;Returns:     (no Return -> mainline program)
;**********************************************************
#make_COM#
; COM file is loaded at CS:0100h
; (the value of CS is set by Operating System)
ORG 100h

        jmp main

TEST1   db      'Hello world how are you'

main:   mov al,'h'              ; Immediate addressing
                                ; puts 'h' into al

        mov bl,al               ; register addressing
                                ; puts what was in AL into BL

        mov al, test1           ; direct (from memory)
                                ; puts the first character of string
                                ; into AL

        hlt
```

Title Block

- Contains info about the program and/or subroutine

Should include:

- Name of the program/routine
- Author
- Date of creation and the date of revision (if applicable)
- Function (what it does)
- Process (how it does the function if not self explanatory)
- On Call (What information needs to be passed to the program/routine and where the information must be located when the program/routine is started)
- Returns (where will the answer be stored)



```
;**********************************************************
;Name:        Addressing Mode Demo

;Author:      Dan Kohn

;Date:        9/12/02

;Function:    To demonstrate the various addressing modes
;             of the 80x86 processor

;On call:     (Self contained code, no inputs needed)

;Returns:     (no Return -> mainline program)
;**********************************************************
#make_COM#
; COM file is loaded at CS:0100h
; (the value of CS is set by Operating System)
ORG 100h

        jmp main

TEST1   db      'Hello world how are you'

main:   mov al,'h'              ; Immediate addressing
                                ; puts 'h' into al

        mov bl,al               ; register addressing
                                ; puts what was in AL into BL

        mov al, test1           ; direct (from memory)
                                ; puts the first character of string
                                ; into AL

        hlt
```

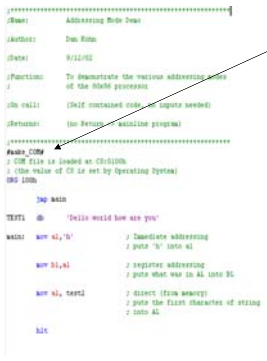
Assembler Directives

The **Output File Type Directives:**

#MAKE_COM#
#MAKE_BIN#
#MAKE_BOOT#
#MAKE_EXE#

You can insert these directives in the source code to specify the required output type for the file.

ORG – Where to place the program in memory



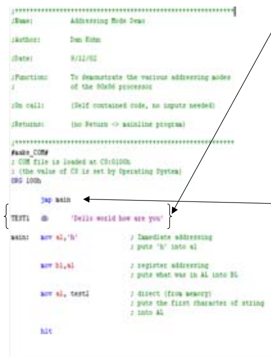
```
;**********************************************************
;Name:           Addressing Mode Demo

;Author:         Dan Kohn

;Date:           9/12/02

;Function:       To demonstrate the various addressing modes
;                of the 80x86 processor

;On call:        (Self contained code, no inputs needed)

;Returns:        (no Return -> mainline program)

;**********************************************************
#make_COM#
; COM file is loaded at CS:0100h
; (the value of CS is set by Operating System)
ORG 100h

        jmp main

TEST1   db      'Dello world how are you'

main:   mov al,'h'              ; Immediate addressing
                                ; puts 'h' into al

        mov bl,al               ; register addressing
                                ; puts what was in AL into BL

        mov al, test1           ; direct (from memory)
                                ; puts the first character of string
                                ; into AL

        hlt
```

Assembler Directives

The **Output File Type Directives:**

#MAKE_COM#
#MAKE_BIN#
#MAKE_BOOT#
#MAKE_EXE#

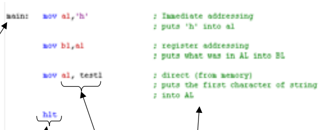
You can insert these directives in the source code to specify the required output type for the file.



```
;**********************************************************
;Name:           Addressing Mode Demo

;Author:         Dan Kohn

;Date:           9/12/02

;Function:       To demonstrate the various addressing modes
;                of the 80x86 processor

;On call:        (Self contained code, no inputs needed)

;Returns:        (no Return -> mainline program)

;**********************************************************
#make_COM#
; COM file is loaded at CS:0100h
; (the value of CS is set by Operating System)
ORG 100h

        jmp main

TEST1   db      'Dello world how are you'

main:   mov al,'h'              ; Immediate addressing
                                ; puts 'h' into al

        mov bl,al               ; register addressing
                                ; puts what was in AL into BL

        mov al, test1           ; direct (from memory)
                                ; puts the first character of string
                                ; into AL

        hlt
```

Variables, arrays and strings

- Programs typically place numeric values, strings and variable space in memory at the top of a program. In this example the string "Dello world how are you" is place in memory to be used later on.
- Because this is stored in memory you must have a JMP instruction to jump over this space and point to the first instruction in the program.



```
main:   mov al,'h'              ; Immediate addressing
                                ; puts 'h' into al

        mov bl,al               ; register addressing
                                ; puts what was in AL into BL

        mov al, test1           ; direct (from memory)
                                ; puts the first character of string
                                ; into AL

        hlt
```

Labels
Point to lines of code (for jump instructions and subroutines)

Operation Codes
ASM instructions (mnemonics)

Operands
Registers, constants, addresses (labels) or variables

Comments
Should describe WHY something is being done (not what it is doing – the code tells you that). Start with ; (semicolon)

```
mov al, testl
```

Destination Operand – where the result ends up

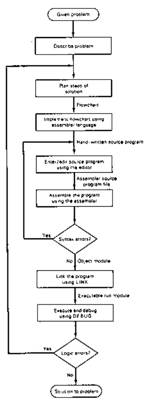
Source Operand

Program Development

Section 3.2

Program Development Cycle

[Fig 3.2 page 66]


Given problem
Describe problem
Plan steps of solution
Flowchart
Implement flowchart using assembler language
Hand-written source program
Enter/edit source program using the editor
Assembler source program file
Assemble the program using the assembler
Yes
Syntax errors?
No
Object module
Link the program using LINK
Executable run module
Execute and debug using DEBUG
Yes
Logic errors?
No
Solution to problem

Step 1 – Describe the Problem

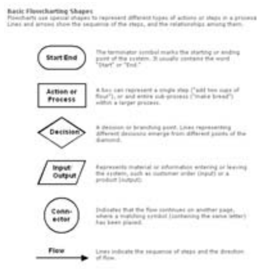
- A programmer cannot program a solution to a problem until he/she understands the problem!
- Divide and conquer! Break larger tasks into smaller ones!

Step 2 - Plan the Solution

- Algorithm - A formula or set of steps for solving a particular problem. To be an algorithm, a set of rules must be unambiguous and have a clear stopping point. Algorithms can be expressed in any language, from natural languages like English to programming languages. [They can also be expressed by using flowcharts].

http://www.pcwebopedia.com

Step 3 - Flowchart



Basic Flowcharting Shapes
Flowcharts use special shapes to represent different types of actions or steps in a process. Lines and arrows show the sequence of the steps, and the relationships among them.

The terminator symbol marks the starting or ending point of the system. It usually contains the word "Start" or "End."

A box can represent a single step ("add two cups of flour"), or and entire sub-process ("make bread") within a larger process.

A decision or branching point. Lines representing different decisions emerge from different points of the diamond.

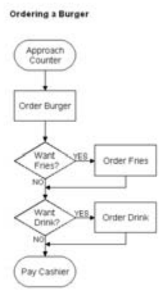
Represents material or information entering or leaving the system, such as customer order (input) or a product (output).

Indicates that the flow continues on another page, where a matching symbol (containing the same letter) has been placed.

Lines indicate the sequence of steps and the direction of flow.

http://www.smartdraw.com/resources/centers/flowcharts/tutorial1.htm

Flowcharting (example)



http://www.smartdraw.com/resources/centers/flowcharts/tutorial1.htm

- Step 4 – Convert flowchart to ASM code
- Step 5 – Enter code into an editor program
- Step 6 – Compile the code
- Step 7 – Debug the syntax (Repeat steps 5-7 until it compiles)
- Step 8 – Debug the logic (repeat steps 5-8 until the program works correctly)