Assembly Language for Intel-Based Computers, 4th Edition

Kip R. Irvine

Chapter 3: Assembly Language Fundamentals

Assembling, Linking and Running Programs

Example Programs

Slides prepared by Kip R. Irvine

Revision date: 09/15/2002

Modified by Dr. Nikolay Metodiev Sirakov
February 2013

- Chapter corrections (Web) Assembly language sources (Web)

Example: Adding and Subtracting Integers

```
TITLE Add and Subtract          (AddSub.asm)
; Program Description: This program adds and subtracts 32-bit integers.
; Author:
; Creation Date:
; Revisions:
; Date:           Modified by:

INCLUDE Irvine32.inc
.code
main PROC

        mov eax,10000h                          ; EAX = 10000h
        add eax,40000h                          ; EAX = 50000h
        sub eax,20000h                          ; EAX = 30000h
        call DumpRegs                           ; display registers
        exit
main ENDP
END main
```

Example Output

Program output, showing registers and flags:

```
EAX=00030000  EBX=7FFDF000  ECX=00000101  EDX=FFFFFFFF
ESI=00000000  EDI=00000000  EBP=0012FFF0  ESP=0012FFC4
EIP=00401024  EFL=00000206  CF=0  SF=0  ZF=0  OF=0
```

Suggested Coding Standards (1 of 2)

- Some approaches to capitalization
  - capitalize nothing
  - capitalize everything
  - capitalize all reserved words, including instruction mnemonics and register names
  - capitalize only directives and operators
- Other suggestions
  - descriptive identifier names
  - spaces surrounding arithmetic operators
  - blank lines between procedures

Suggested Coding Standards (2 of 2)

- Indentation and spacing
  - code and data labels – no indentation
  - executable instructions – indent 4-5 spaces
  - comments: begin at column 40-45, aligned vertically
  - 1-3 spaces between instruction and its operands
    - ex: `mov  ax,bx`
  - 1-2 blank lines between procedures

Alternative Version of AddSub

```
TITLE Add and Subtract                    (AddSubAlt.asm)

; This program adds and subtracts 32-bit integers.
.386
.MODEL flat,stdcall
.STACK 4096

ExitProcess PROTO, dwExitCode:DWORD
DumpRegs PROTO

.code
main PROC
    mov eax,10000h            ; EAX = 10000h
    add eax,40000h            ; EAX = 50000h
    sub eax,20000h            ; EAX = 30000h
    call DumpRegs
    INVOKE ExitProcess,0
main ENDP
ExitProcess
END main
```

Program Template

```
TITLE Program Template                  (Template.asm)

; Program Description:
; Author:
; Creation Date:
; Revisions:
; Date:                    Modified by:

INCLUDE Irvine32.inc
.data
    ; (insert variables here)
.code
main PROC
    ; (insert executable instructions here)
    exit
main ENDP
    ; (insert additional procedures here)
END main
```

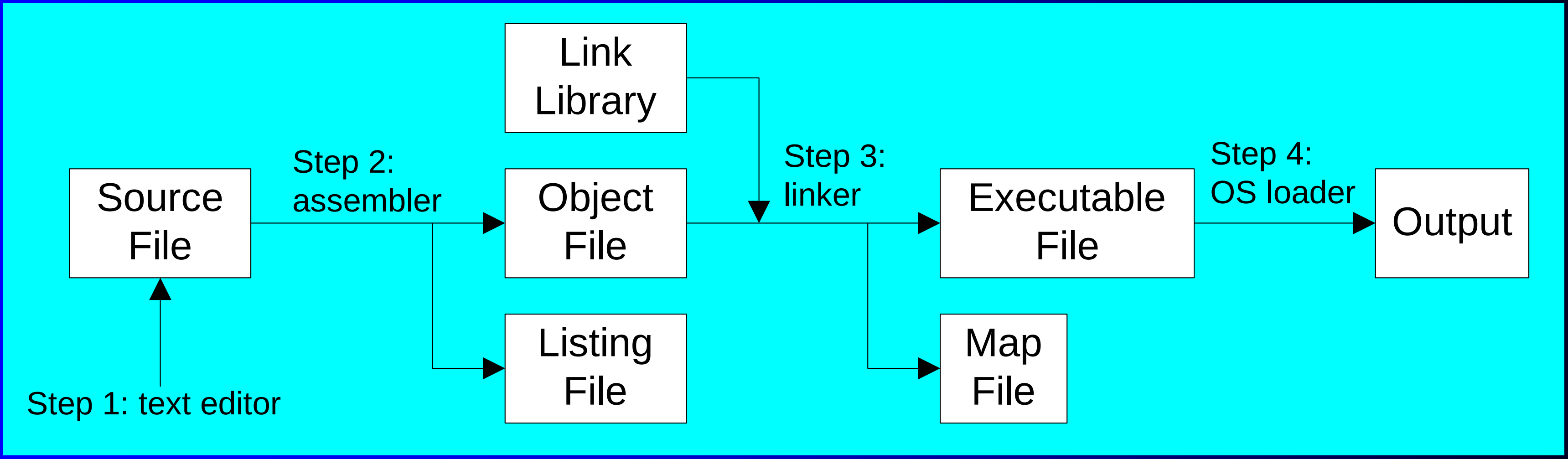
Instructors: please customize as needed

Assembling, Linking, and Running Programs

- Assemble-Link-Execute Cycle
- make32.bat
- Listing File
- Map File

Assemble-Link Execute Cycle

- The following diagram describes the steps from creating a source program through executing the compiled program.
- If the source code is modified, Steps 2 through 4 must be repeated.



make32.bat

- Called a batch file
- Run it to assemble and link programs
- Contains a command that executes ML.EXE (the Microsoft Assembler)
- Contains a command that executes LINK32.EXE (the 32-bit Microsoft Linker)
- Command-Line syntax:

  make32 *progName*

  (*progName* does not include the .asm extension)

Use make16.bat to assemble and link Real-mode programs

Listing File

- Use it to see how your program is compiled
- Contains
    - source code
    - addresses
    - object code (machine language)
    - segment names
    - symbols (variables, procedures, and constants)
- Example: addSub.lst

Listing File, AddSub

- Listing File_AddSubC – AddSubC.LST
- Listing File_AddSub - AddSub.LST
- Listing File_AddSub32 – AddSub32.LST

Map File

- Information about each program segment:
  - starting address
  - ending address
  - size
  - segment type
- Example: addSub.map

Add and Subtract, 16-Bit Version, Variables

```
TITLE Add and Subtract, Version 2        (AddSub2.asm)
INCLUDE Irvine16.inc
.data
val1 DWORD 10000h
val2 DWORD 40000h
val3 DWORD 20000h
finalVal DWORD ?
.code
main PROC
    mov ax,@data                      ; initialize DS
    mov ds,ax
    mov eax,val1                      ; get first value
    add eax,val2                      ; add second value
    sub eax,val3                      ; subtract third value
    mov finalVal,eax                  ; store the result
    call DumpRegs                     ; display registers
    exit
main ENDP
END main
```

Add and Subtract, 32-Bit Version, Variables

```
•    TITLE Add and Subtract, Version 2      (AddSub2.asm)

•    INCLUDE Irvine32.inc
•    .data
•    val1 DWORD 10000h
•    val2 DWORD 40000h
•    val3 DWORD 20000h
•    finalVal DWORD ?

•    .code
•    main PROC
•            mov eax,val1          ; get first value
•            add eax,val2          ; add second value
•            sub eax,val3          ; subtract third value
•            mov finalVal,eax      ; store the result
•            call DumpRegs         ; display registers
•            exit
•    main ENDP
•    END main
```

Redirecting Input-Output (2 of 2)

- Standard input, standard output can both be redirected
- Standard error cannot be redirected
- Suppose we have created a program named myprog.exe that reads from standard input and writes to standard output. Following are MS-DOS commands that demonstrate various types of redirection:

```
myprog < infile.txt

myprog > outfile.txt

myprog < infile.txt > outfile.txt
```

INT Instruction

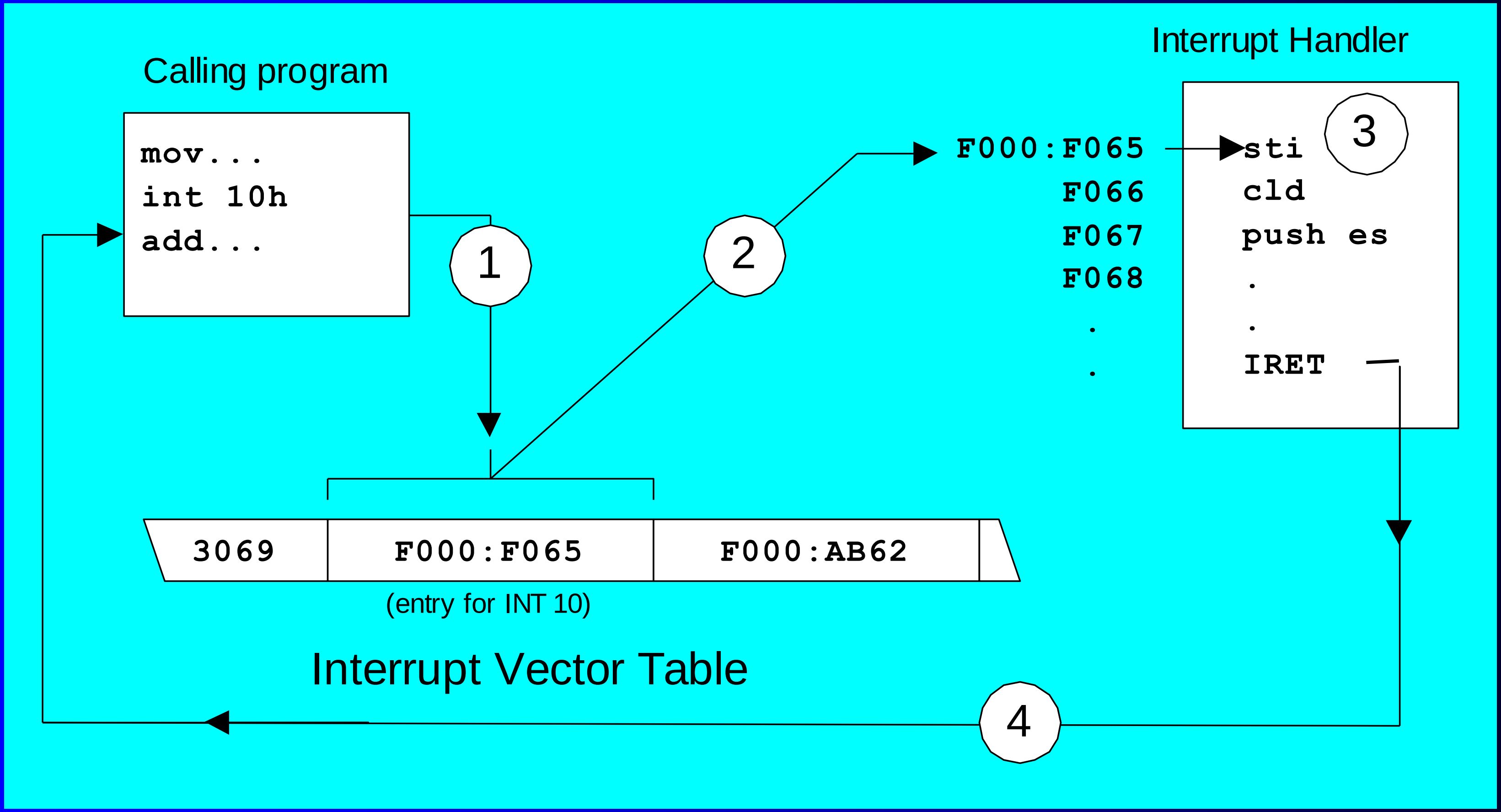
- The INT instruction executes a software interrupt.
- The code that handles the interrupt is called an interrupt handler.
- Syntax:

```
INT number
(number = 0..FFh)
```

The Interrupt Vector Table (IVT) holds a 32-bit segment-offset address for each possible interrupt handler.

Interrupt Service Routine (ISR) is another name for interrupt handler.

Interrupt Vectoring Process



Common Interrupts

- INT 10h Video Services
- INT 16h Keyboard Services
- INT 17h Printer Services
- INT 1Ah Time of Day
- INT 1Ch User Timer Interrupt
- INT 21h MS-DOS Services

MS-DOS Function Calls (INT 21h)
Page 465

- ASCII Control Characters
- Selected Output Functions
- Selected Input Functions
- Example: String Encryption
- Date/Time Functions

ASCII Control Characters

Many INT 21h functions act upon the following control characters:

- 08h - Backspace (moves one column to the left)
- 09h - Horizontal tab (skips forward n columns)
- 0Ah - Line feed (moves to next output line)
- 0Ch - Form feed (moves to next printer page)
- 0Dh - Carriage return (moves to leftmost output column)
- 1Bh - Escape character

INT 21h Function 09h:
Write String to Standard Output

- The string must be terminated by a '$' character.
- DS must point to the string's segment, and DX must contain the string's offset:

```
.data
string BYTE "This is a string$"

.code
mov   ah,9
mov   dx,OFFSET string
int   21h
```

Selected Input Functions

- 01h, 06h - Read character from standard input
- 0Ah - Read array of buffered characters from standard input
- 0Bh - Get status of the standard input buffer
- 3Fh - Read from file or device

INT 21h Function 0Ah:

Read buffered array from standard input –**Page 469**

- Requires a predefined structure to be set up that describes the maximum input size and holds the input characters.
- Example:

```
count = 80

KEYBOARD STRUCT
    maxInput BYTE count                ; max chars to input
    inputCount BYTE ?                  ; actual input count
    buffer BYTE count DUP(?)           ; holds input chars
KEYBOARD ENDS
```

INT 21h Function 0Ah

Executing the interrupt:

```
.data
kybdData KEYBOARD <>

.code
   mov ah,0Ah
   mov dx,OFFSET kybdData
   int 21h
```

INT 21h Function 2Ah:
Get system date

- Returns year in CX, month in DH, day in DL, and day of week in AL

```
mov   ah,2Ah
int   21h
mov   year,cx
mov   month,dh
mov   day,dl
mov   dayOfWeek,al
```

INT 21h Function 2Ch:
Get system time

- Returns hours (0-23) in CH, minutes (0-59) in CL, and seconds (0-59) in DH, and hundredths (0-99) in DL.

```
mov   ah,2Ch
int   21h
mov   hours,ch
mov   minutes,cl
mov   seconds,dh
```

Example Programs

- TITLE Display the Date and Time
- Read, display, and copy a text file