

Week 3

Chapter 3: Assembly Language Fundamentals

Class 7

Chapter Overview

- Basic Elements of Assembly Language
- Example: Adding and Subtracting Integers
- Assembling, Linking, and Running Programs
- Defining Data
- Symbolic Constants
- 64-Bit Programming

Basic Elements of Assembly Language

- Integer constants
- Integer expressions
- Character and string constants
- Reserved words and identifiers
- Directives and instructions
- Labels
- Mnemonics and Operands
- Comments
- Examples

Integer Constants

- Optional leading + or – sign
- binary, decimal, hexadecimal, or octal digits
- Common radix characters:
 - h – hexadecimal
 - d – decimal
 - b – binary
 - r – encoded real

Examples: 30d, 6Ah, 42, 1101b

Hexadecimal beginning with letter: 0A5h

Integer Expressions

- Operators and precedence levels:

Operator	Name	Precedence Level
()	parentheses	1
+, -	unary plus, minus	2
*, /	multiply, divide	3
MOD	modulus	3
+, -	add, subtract	4

- Examples:

Expression	Value
16 / 5	3
-(3 + 4) * (6 - 1)	-35
-3 + 4 * 6 - 1	20
25 mod 3	1

Character and String Constants

- Enclose character in single or double quotes
 - 'A', "x"
 - ASCII character = 1 byte
- Enclose strings in single or double quotes
 - "ABC"
 - 'xyz'
 - Each character occupies a single byte
- Embedded quotes:
 - 'Say "Goodnight," Gracie'

Reserved Words and Identifiers

- Reserved words cannot be used as identifiers
 - Instruction mnemonics, directives, type attributes, operators, predefined symbols
 - See MASM reference in Appendix A
- Identifiers
 - 1-247 characters, including digits
 - **not** case sensitive
 - first character must be a letter, _, @, ?, or \$

Directives

- Commands that are recognized and acted upon by the assembler
 - Not part of the Intel instruction set
 - Used to declare code, data areas, select memory model, declare procedures, etc.
 - not case sensitive
- Different assemblers have different directives
 - NASM not the same as MASM, for example

Instructions

- Assembled into machine code by assembler
- Executed at runtime by the CPU
- We use the Intel IA-32 instruction set
- An instruction contains:
 - Label (optional)
 - Mnemonic (required)
 - Operand (depends on the instruction)
 - Comment (optional)

Read topic "x86 Instruction Encoding" from Page number 539 to 547

Labels

- Act as place markers
 - marks the address (offset) of code and data
- Follow identifier rules
- Data label
 - must be unique
 - example: **myArray** (not followed by colon)
- Code label
 - target of jump and loop instructions
 - example: **L1:** (followed by colon)

Mnemonics and Operands

- Instruction Mnemonics
 - memory aid
 - examples: MOV, ADD, SUB, MUL, INC, DEC
- Operands
 - constant
 - constant expression
 - register
 - memory (data label)

Constants and constant expressions are often called
immediate values

Comments

- Comments are good!
 - explain the program's purpose
 - when it was written, and by whom
 - revision information
 - tricky coding techniques
 - application-specific explanations
- Single-line comments
 - begin with semicolon (;)
- Multi-line comments
 - begin with COMMENT directive and a programmer-chosen character
 - end with the same programmer-chosen character

Instruction Format Examples

- No operands
 - `stc` ; set Carry flag
- One operand
 - `inc eax` ; register
 - `inc myByte` ; memory
- Two operands
 - `add ebx,ecx` ; register, register
 - `sub myByte,25` ; memory, constant
 - `add eax,36 * 25` ; register, constant-expression

What's Next

- Basic Elements of Assembly Language
- **Example: Adding and Subtracting Integers**
- Assembling, Linking, and Running Programs
- Defining Data
- Symbolic Constants
- 64-Bit Programming

Example: Adding and Subtracting Integers

```
; AddTwo.asm - adds two 32-bit integers
```

```
.386
```

```
.model flat,stdcall
```

```
.stack 4096
```

```
ExitProcess PROTO, dwExitCode:DWORD
```

```
.code
```

```
main PROC
```

```
    mov    eax,5        ; move 5 to the EAX register
```

```
    add    eax,6        ; add 6 to the EAX register
```

```
    INVOKE ExitProcess,0
```

```
main ENDP
```

```
END main
```

Example Output

Showing registers and flags in the debugger:

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: right side of page, aligned vertically
 - 1-3 spaces between instruction and its operands
 - ex: `mov ax,bx`
 - 1-2 blank lines between procedures

Program Template

```
; Program Template                (Template.asm)

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

.386
.model flat,stdcall
.stack 4096
ExitProcess PROTO, dwExitCode:DWORD
.data
; declare variables here
.code
main PROC
    ; write your code here
    INVOKE ExitProcess,0
main ENDP
; (insert additional procedures here)
END main
```