Introduction to Assembly Language

Muhammad Afzaal
m.afzaal@nu.edu.pk

Book Chapter

- “Assembly Language for x86 Processors”
- Author “Kip R. Irvine”
- 6th Edition
- Chapter 3
  - Section 3.1

Basic Elements of Assembly Language

- Integer Constants
- Integer Expressions
- Real Number Constants
- Character Constants
- String Constants
- Reserved Words
- Identifiers
- Directives
- Instructions
- The NOP (No OPeration) Instruction

Integer Constants (1/2)

- Represented by an optional leading sign, one or more digits and an optional suffix
- Suffix is also called radix and represents the base of number
- If not radix is given, then number is decimal
- A byte can hold an integer value in the range
  - 0 → 255 in case of unsigned number
  - -128 → +127 in case of signed numbers

```
[{+|-}] digits {radix}
```

Integer Constants (2/2)

- `Radix` may be one of the following
  - `h` for Hexadecimal
  - `d` or decimal
  - `q/o` for octal
  - `b` for binary
- Examples
  - `26`, `26d` are decimal number
  - `1100110b` is binary
  - `45q`, `45o` are octal
  - `1Ah`, `0A1h` are hexadecimal

Integer Expressions (1/2)

- A mathematical expression involving integer values and arithmetic operators
- These expressions can be evaluated only at assembly time
  - No any runtime expression

Integer Expressions (2/2)

- Different operators have different precedence
  - Precedence is the order of execution when two or more operators appear in the same expression
- Use parentheses in order to not confuse with precedence

Operator	Name	Precedence Level
()	Parentheses	1
+, -	Unary plus, minus	2
*, /	Multiply, Divide	3
MOD	Modulus	3
+, -	Add, Subtract	4

Real Number Constants

- Decimal Real contains an optional sign followed by an integer, a decimal point, an option integer and an optional exponent

  `[{+|-}]` *`integer`*`.[`*`integer`*`][`*`exponent`*`]c`

  - *`Exponent`* `E[{+,-}]`*`integer`*

  - Examples are
    - +5.0
    - 2.
    - -23.87E+04
    - 35.E7

Character Constants

- A single character enclosed in single or double quotes
- Each character is stored as a byte
- Examples are
  - `‘Y12’`
  - `“d”`

String Constants

- Sequence of characters including spaces
- Enclosed in single or double quotes
- Examples are
  - `‘Hello’`
  - `‘7865’`
  - `“Hello World”`
- Embedded quotes can be used if in proper order
- Examples are
  - `“This isn’t a test”`
  - `‘Say “Good night” to him’`

Reserved Words

- Have special meaning and can only be used in correct context
  - Instruction mnemonics like MOV, ADD, SUB, INT etc.
  - Register Names like AX, BX, DL, DH etc.
  - Directives like .DATA, .CODE etc.
  - Attributes like BYTE, WORD etc.
  - Operators used in constant expressions
  - Predefined symbols

Identifiers

- Name of a variable, constant, procedure or a code label selected by programmer
- Some rules to follow while choosing identifier names
  - From 1 to 247 number of characters
  - Names are not case sensitive
  - An identifier cannot be the same as an assembler reserved word
  - First character must be a letter (a-z, A-Z), underscore(_), @, ? Or $. Subsequent characters may also contain digits
- Examples are `var1`, `CounT`, `_name`, `_1344`

Directives

- A command embedded in the source code that is recognized and acted upon by assembler
- Directives can define variables and procedures
- They assign names to memory segments
- Not case sensitive
- Examples are
  - `DWORD`
  - `.data`
  - `.code`

Instructions

- A statement that becomes executable when a program is assembled
- Translated by assembler into machine language
- An Instruction contains four basic parts
  - Label (optional)
  - Instruction Mnemonic (required)
  - Operand(s) (usually required)
  - Comment (optional)
- Basic syntax is

```
[label:] mnemonic [operand] [;comment]
```

Label

- An identifier that acts as a place-marker for instructions and data
- A label placed just before an instruction/variable implies its address
- Data Labels
  - Name of a variable
- Code Labels
  - Must end with a colon (:)
  - Used as targets in jump and `loop` instructions

Instruction Mnemonics

- A short word that identifies an instruction
- A mnemonic is a device that assists memory
- Assembly language instruction mnemonics provide hints about the type of operation they perform
  - `MOV` assigns one value to other
  - `ADD` adds two values
  - `SUB` subtracts two values
  - `JMP` jumps to a new location

Operands

- Instructions can have 0 – 3 operands
- Operand can be a register, memory operand, constant expression or an I/O port
  - Memory operand is specified either by variable name or by registers containing variable address
- Instruction with 0 operand → `STC`
- Instruction with 1 operand → `INC AX`
- Instruction with 2 operand → `MOV AX, BX`
- Instruction with 3 operand → `IMUL AX, BX, 4`

Comments (1/2)

- Comments can be used to inform the code reader about the design of code
- A program typically contains the following information at the top of the program
  - Description of the program’s purpose
  - Programmers involved
  - Program creation and revision dates
  - Technical notes about the program’s implementation

Comments (2/2)

- Single line comments
  - Begin with a semicolon (`;`) character
  - All character after the semicolon on the same line are ignored by the assembler
  - `STC ; set carry flag`
- Block comments
  - Begin with assembly language directive COMMENT and a user specified symbol

  -
    ```
    COMMENT !
     this is a comment
     this is also a comment
     !
    ```

NOP (No Operation) Instruction

- The safest and even most useless instruction in assembly language
- Does not do anything except occupying 1 byte of program storage
- Sometimes used by assemblers to align code to even-address boundaries

```
00000000 66 8B C3 MOV AX, BX
00000003 90       NOP
00000004 8B D1    MOV EDX, ECX
```