ASSEMBLY LANGUAGE FUNDAMENTALS

MODULE OUTLINE

1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE
2. ADDING AND SUBTRACTING INTEGERS
3. ASSEMBLING, LINKING AND RUNNING PROGRAMS
4. DEFINING DATA
5. SYMBOLIC CONSTANTS
6. REAL-ADDRESS MODE PROGRAMMING

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 01

5.1.1. INTEGER CONSTANTS

- **Radix**
  - **h, Hexadecimal**
  - **q/o, Octal**
  - **d, Decimal**
  - **b, binary**
  - **r, Encoded Real**
  - **t, Decimal (alternate)**
  - **y, Binary (alternate)**
- **Put a leading zero for an hexadecimal constant beginning with letter.**

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 02

5.1.2. INTEGER EXPRESSIONS

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

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 03

5.1.3. REAL NUMBER CONSTANTS

- **[sign]integer.[integer][exponent]**
- **Examples:**
  - **2.**
  - **+3.0**
  - **-44.2E+05**
  - **26.E5**
- **Encoded Reals:**
  - **0011 1111 1000 0000 0000 0000 0000 0000b**
  - **3F800000r**

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 04

5.1.4. CHARACTER CONSTANTS

- **NASM stores characters in ASCII code**
- **Examples:**
  - **'A'**
  - **"d“**

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 05

5.1.5. STRING CONSTANTS

- **Examples:**
  - **'ABC'**
  - **'X'**
  - **"Good night, Gracie"**
  - **'4096‘**
- **Embedded quotes are permitted**
  - **"This isn't a test"**
  - **'Say "Good night," Uvulwa'**

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 06

5.1.6. RESERVED DWS

- **Instructions: MOV, ADD, MUL**
- **Registers Names: EAX, ECX, CS, SS**
- **Directives: ORG (origin), DD (double word)**
- **Attributes: DB, DW**
- **Operators: +, -, *, /, etc**
- **Predefined symbols: @**

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 07

5.1.7. IDENTIFIERS

- **May contain 1 to 247 characters**
- **Case sensitive**
- **First character (A…Z, a…z) or _, @, ? or $**
- **Do not use Reserved words**

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 08

5.1.8. DIRECTIVES

- **Commands instructing the assembler**
- **Do not execute at runtime**
- **Can define variables, macros and procedures**
- **Assign names to memory**
- **NASM is case sensitive;**
  - **e.g.: .DATA, .data and .Data are not equivalent**

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 09

5.1.9. INSTRUCTIONS

- **Have four basic parts:**
  - **Label (optional)**
  - **Instruction mnemonic (required)**
  - **Operand(s) (usually required)**
  - **Comment (optional)**
- **Basic syntax:**
  - ***[label:] mnemonic[operands] [;comment]***

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 10

5.1.9. INSTRUCTIONS - contd

- **Label is a place marker**
  - ***Data Labels***
    - **e.g.:**
```
count: DD 100 ; create a variable ‘count’
array: DD 1024, 2048, 4096, 8192    ; multiple data items
```
  - ***Code Labels***
    - **e.g1.:**
```
target:
        mov     ax, bx
        …
        jmp     target
```
    - **e.g2:**
```
label1: mov     ax, bx  ; same line with
                        ; instruction
```

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 11

5.1.9. INSTRUCTIONS - contd

- **Instruction Mnemonic**
  - `mov  ;Move (assign) one value to another`
  - `add  ;Add two values`
  - `sub  ;Subtract one value from another`
  - `mul  ;Multiply two values`
  - `jmp  ;Jump to a new location`
  - `call ;Call a procedure`

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 12

5.1.9. INSTRUCTIONS - contd

- **Operands**
  - `stc ;Set carry flag – no operand`
  - `inc eax ;Add 1 to EAX`
  - `mov count, ebx ;Move EBX to count`
  - `imul eax, ebx, 5 ;Store (5*EBX) in EAX`

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 13

5.1.9. INSTRUCTIONS - contd

- **Comments**
  - ***Program's purpose***
  - ***Author or reviser***
  - ***Program date stamp***
  - ***Technical notes***
- ***Multi-line comments***
  - **e.g.:**

```
COMMENT !
            This line is a comment.
            This line is also a comment.
      !
```

5.1. BASIC ELEMENTS OF ASSEMBLY LANGUAGE - 14

5.1.10.THE NOP (No Operation) INSTRUCTION

- Takes up one byte of storage
- Sometimes used by assemblers/compilers to align code to even-address
  - **e.g.:**

```
00000000 66 8B C3 mov ax,bx
00000003 90 nop                ; align next instruction
00000004 8B D1 mov edx,ecx
```

5.2. ADDING AND SUBTRACTING INTEGERS - 01

5.2.1. EXAMPLE OF ADD & SUBTRACT

```
; This program adds and subtracts 32-bit integers.
.386
.model flat,stdcall
.stack 4096

.text
main:
     mov eax,10000h                          ; EAX = 10000h
     add eax,40000h                          ; EAX = 50000h
     sub eax,20000h                          ; EAX = 30000h
     call DumpRegs
     INVOKE ExitProcess,0
     ret

proc ExitProcess
     …
endproc

proc DumpRegs
     …
endproc
```

5.2. ADDING AND SUBTRACTING INTEGERS - 02

5.2.1. PROGRAM TEMPLATE

```
TITLE Program Template (Template.asm)
; Program Description:
; Author:
; Creation Date:
; Revisions:
; Date:
%include “Along32.inc”
.data
    ; (insert variables here)
.text
main:
    ; (insert executable instructions here)
    ; (insert additional procedures here)
    ret
```

5.3. ASSEMBLING, LINKING AND RUNNING PROGRAMS

5.3.1. THE ASSEMBLE-LINK-EXECUTE CYCLE

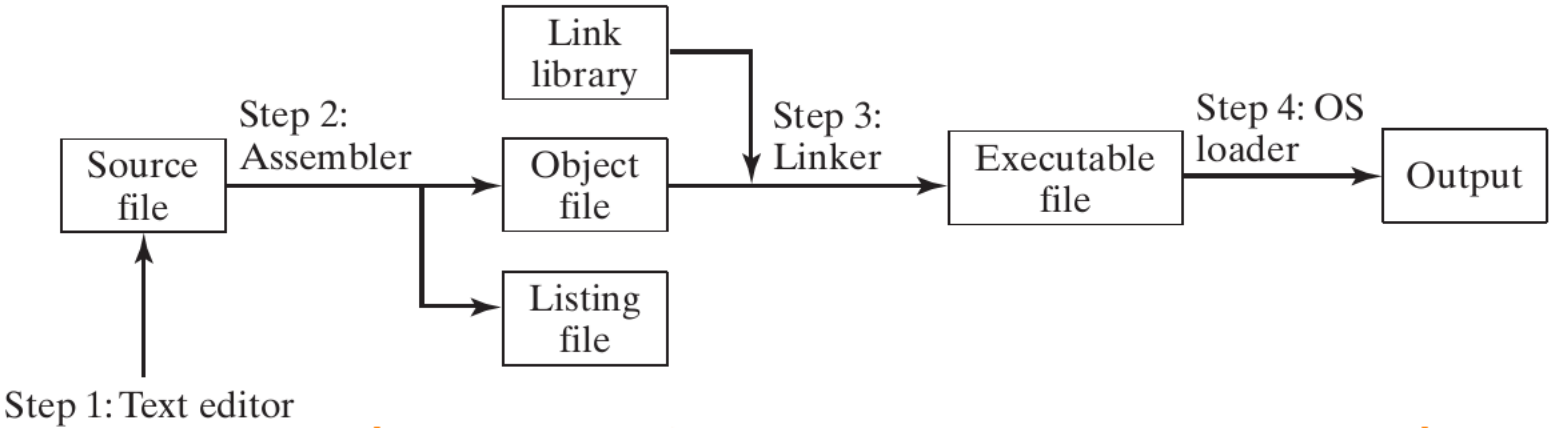

Figure 5.3.1.1: Assemble-Link-Execute Cycle

5.4. DEFINING DATA - 01

5.4.1. INTRINSIC DATA TYPES

Data Type	Width (bits)	Precision (bits)	Range	NASM Equivalent
Unsigned BYTE	8	8	0 TO 255	DB
Unsigned WORD	16	16	0 TO 65,535	DW
Unsigned DWORD	32	32	0 TO 4,294,967,295	DD
Unsigned QWORD	64	64	0 TO 2^{64}	DQ
Unsigned DQWORD	128	128	0 to 2^{128}	DO
Signed BYTE	8	7	-128 to 127	DB
Signed WORD	16	15	-32,768 to 32,767	DW
Signed DWORD	32	31		DD
Signed QWORD	64	63	9.22×10^{-17} to 9.22×10^{17}	DQ
Single Precision Floating-Point	32	24	8.43×10^{-37} to 3.40×10^{38}	DD
Double Precision Floating-Point	64	53	4.19×10^{-307} to 1.79×10^{308}	DQ
Double Extended Precision Floating-Point	80	64	3.37×10^{-4932} to 1.18×10^{4932}	DT

Table 5.4.1.1: Intrinsic Data Types

5.4. DEFINING DATA - 02

5.4.2. DATA DEFINITION STATEMENT

- `[name] directive initializer [,initializer]...`
- `e.g.:`

```
count: DD 12345
```

Table 5.4.1.1: Intrinsic Data Types

5.4. DEFINING DATA - 03

5.4.3. DEFINING DB DATA

```
value1: DB 'A'        ; character constant
Value2: DB 0          ; smallest unsigned byte
Value3: DB 255        ; largest unsigned byte
Value4: DB –128       ; smallest signed byte
Value5: DB +127       ; largest signed byte
```

5.4. DEFINING DATA - 04

5.4.3. DEFINING DB DATA – contd

- ***DB Directive***

```
val1: DB 255     ; unsigned byte
val2: DB -128    ; signed byte
```

- ***Multiple Initialisers***

```
list: DB 10, 20, 30, 40
```

5.4. DEFINING DATA - 05

5.4.3. DEFINING DB DATA – contd

- ***Multiple Initialisers***

```
list: DB 10, 20, 30, 40
```

Offset	Value
0000:	10
0001:	20
0002:	30
0003:	40

Figure 5.4.3.1: Memory Layout of a Byte Sequence

5.4. DEFINING DATA - 06

5.4.3. DEFINING DB DATA – contd

- ***Multiple Initialisers (contd)***
  - ***array***

```
list: DB 10,20,30,40
      DB 50,60,70,80
      DB 81,82,83,84
```

  - ***Using different radixes***

```
list1: DB 10, 32, 41h, 00100010b
list2: DB 0Ah, 20h, 'A', 22h
```

5.4. DEFINING DATA - 07

5.4.3. DEFINING DB DATA – contd

- ***Defining Strings***

```
greeting1: DB "Good afternoon",0
greeting2: DB 'Good night',0
greeting1: DB "Welcome to the Encryption Demo program"
           DB "created by Kiva Makangila",0dh,0ah
           DB "If you wish to modify this program, please"
           DB "send me a copy.",0dh,0ah,0
```

5.4. DEFINING DATA - 08

5.4.3. DEFINING DB DATA – contd

- ***TIMES Operator* - allocates storage for multiple data items**

```
TIMES DB 20 0        ; 20 bytes, all equal to zero
TIMES DB 20          ; 20 bytes, uninitialized
TIMES DB 4 "STACK"   ; 4 bytes: "STACKSTACKSTACKSTACK"
```

5.4. DEFINING DATA - 09

5.4.4. DEFINING DW DATA

- **DW directives to create one or more 16-bit integers**
- e.g.:

```
word1: DW 65535   ;largest unsigned value
word2: DW -32768  ;smallest signed value
word3: DW         ;uninitialized, unsigned
```

=> Legacy DW directive

```
val1: DW 65535     ;unsigned
val2: DW -32768    ;signed
```

5.4. DEFINING DATA - 10

5.4.4. DEFINING DW DATA - contd

- **ARRAYS OF WORDS**

```
myList: DW 1, 2, 3, 4, 5
```

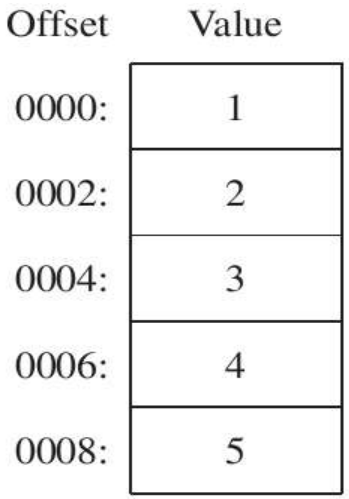

Figure 5.4.2.1: Memory Layout, 16-bit Word array

- **TIMES Operator is more convenient**

```
array: TIMES DW 5  ; 5 values, uninitialised
```

5.4. DEFINING DATA - 11

5.4.5. DEFINING DD DATA

- **DD directives to create one or more 32-bit integers**
- e.g.:

```
val1: DD 12345678h          ;unsigned
val2: DD –2147483648        ;signed
val3: TIMES DD 20           ;unsigned array
```

=> Legacy DD directive

```
val1: DD 12345678h          ;unsigned
val2: DD –2147483648        ;signed
```

5.4. DEFINING DATA - 12

5.4.5. DEFINING DD DATA - contd

- **ARRAYS OF DOUBLEWORDs**

```
myList: DD 1, 2, 3, 4, 5
```

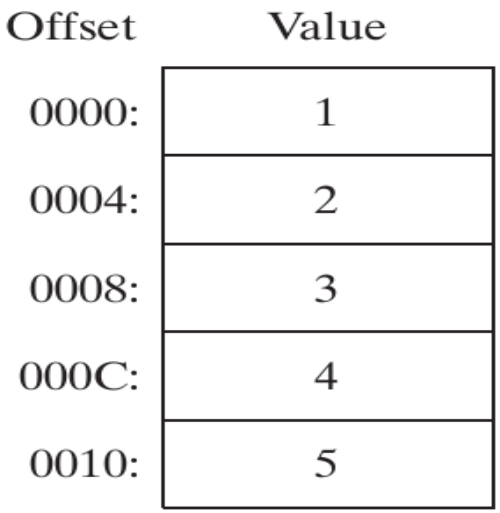

Figure 5.4.2.2: Memory Layout, 32-bit Doubleword Array

5.4. DEFINING DATA - 13

5.4.6. DEFINING PACKET BINARY CODED DECIMAL (BCD, TDB) DATA

- **Intel stores BCD in 10-byte package**

Decimal Value	Storage Bytes
+1234	34 12 00 00 00 00 00 00 00 00
−1234	34 12 00 00 00 00 00 00 00 80