CMPE-310

Lab01- Introduction To NASM

Outline

- Assembly Program Structure
- Learn how to assemble code
- Basic NASM Syntax
- Sample NASM Source Code

Lab policy

NO Sharing Code/Cheating
Some collaboration is okay, too much collaboration is NOT.
We will check your code.

You will upload your code and all home work document reports to submit directory
Class name CMPE_310

More on how to use **submit** is available from here.
Submit instructions will be posted on the course website.

Do NOT save anything on lab computers (ITE375), files get deleted after powering off.

Lab Grading:
30% Demo, 70% Correctness (Functionality)

UTA and TA Contact info

UTA:

Nathaniel Sokolow: nsokolo1@umbc.edu

Simon Rupp: srupp1@umbc.edu

TA :

Sravani Varanasi: sravani1@umbc.edu

Structure of an Assembly Language Statement

General structure of an assembly language statement

LABEL: INSTRUCTION ;COMMENT

Label — address identifier for the statement

Instruction — the operation to be performed

Comment — documents the purpose of the statement

Example:

```
START:  mov  AX, BX     ;Copy the content of BX into AX
```

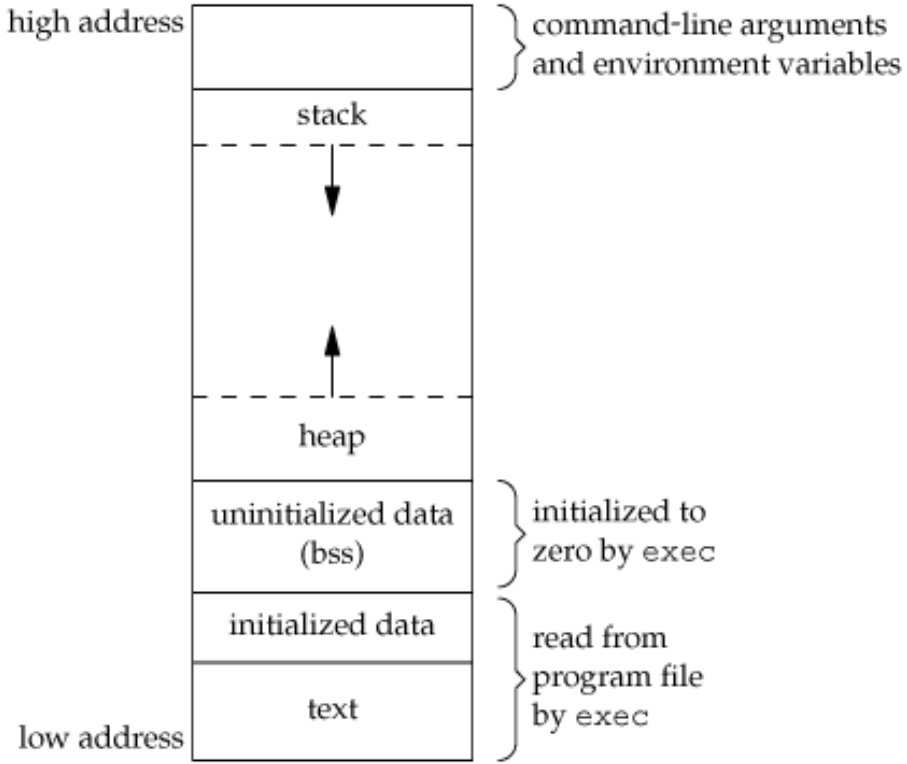
Other examples:

```
        INC   SI        ;Update pointer
        ADD  AX, BX     ;Add the content of BX to AX  and store the result in AX
```

Few instructions have a label — usually marks a jump to point

Not all instructions need a comment

Program memory layout (sections)



Source: http://i.stack.imgur.com/1Yz9K.gif

NASM Program Sections

NASM – Netwide assembler

Documentation on NASM is available from here

The **data section** is used for declaring initialized data or constants.

```
section .data
        msg db "Hello World!", 0xA    ; our string terminated by newline
        len equ $-msg                 ; length of our string
                                      ; equ defines len to a constant
```

The **bss section** is used for declaring variables or uninitialized data.

```
section .bss
        Buffer: resb 256              ; reserve 256 bytes
```

The **text section** is used for keeping the actual code.

```
section .text
        your code here
```

NASM Labels

Labels- Give structure to code and provides target for jump instructions

label: instruction operand

- The ':' is optional, which can cause problems if, for example, you misspell an instruction
- Valid characters in labels are letters, numbers, _, $, #, @, ~, ., and ?.
- Identifier valid starting characters include letters, . , _ and ?.

Local Labels- begin with a '.' and are associated with previous non-local label.

```
label1                  ; some code
        .loop           ; some more code
        jne  .loop      ; jump to previous loop
        ret             ; treated as label1.loop
label2 .loop            ; some more code
        jne  .loop      ; jump to previous loop
```

NASM Compilation

To get command line help, type:

```
nasm -h
```

To compile into an ELF object file .o, type:

```
nasm -f elf myfile.asm
```

To create a listing file, type:

```
nasm -f elf myfile.asm -l myfile.lst
```

To send errors to a file, type: (This option is now deprecated)

```
~~nasm -E myfile.err -f elf myfile.asm~~
```

To include other search paths such as /usr/include, type:

```
nasm -I/usr/include -f elf myfile.asm
```

To include other files in a source file, use:

```
%include "myinc.inc"
```

Content adapted from Dr. Chintan Patel's slides

Assembler and the source program

- Assembly language program (.asm) file—known as source code
- Converted to machine code by a process called assembling
- Machine (object) code output by assembler needs to be linked and loaded before execution
- Source listing output in (.lst) file—printed and used during execution and debugging of program

Debugger – GDB (GNU Debugger)

- Permits programs to be assembled and disassembled
- Line-by-line assembly is possible
- Also permits program to be executed and tested

The Listing file

Listing file contents

- Size of code segment and stack
- Names, types, and values of constants and variables
- # lines and symbols used in the program
- # errors that occurred during assembly

nasm-f elf myfile.asm -l myfile.lst

Hello World

```
;
; Assemble using NASM
;

section .data                   ; section declaration
msg     db  'Hello, world!',0xA ; our string
len     equ $ -                 ; length of our string

section    .text                ; section declaration
global     _start               ; must be declared for linker (ld)

_start:                         ; tell linker entry point

   mov     eax,4                ; system call number (sys_write)
   mov     ebx,1                ; file descriptor (stdout)
   mov     ecx,msg              ; message to write
   mov     edx,len              ; message length
   int     0x80                 ; call kernel

                                ; final exit
   mov     eax,1                ; system call number (sys_exit)
   xor     ebx,ebx              ; sys_exit return status
   int     0x80                 ; call kernel
```

Hello World

Produce hello.o object file:

nasm –f elf hello.asm –l hello.lst

Produce hello ELF executable (Link and Load):

ld -m elf_i386 -s hello.o -o hello

Run the program:

./hello