Basic Assembly

Ned Nedialkov

McMaster University
Canada

SE 3F03
January 2014

Outline

Assemblers

An assembler is a program that translates an assembly language program into binary code

- NASM Netwide Assembler
- TASM Turbo Assembler (Boorland)
- MASM Microsoft Assembler
- . . .

We study NASM

Basic instructions

- **add** `dest, source`
  - `dest=dest+source`
  - `dest` register or memory location
  - `source` register, memory location, immediate
- **sub** `dest, source`
  - `dest=dest-source`
- **mov** `dest, source`
  - `dest <-- source`
  - `dest` register or memory
  - `source` register, memory, or immediate
  - both cannot be memory

Assembly program structure

```
%include "asm_io.inc"
segment .data
; initialized data
segment .bss
; uninitialized data
segment .text
        global asm_main
asm_main:
        enter 0,0               ; setup
        pusha                   ; save all registers
        ; put your code here
        popa                    ; restore all registers
        mov eax, 0              ; return value
        leave
        ret
```

I/O

- C: I/O done through the standard C library
- Assembly: I/O through the standard C library
  ```
  %include "asm_io.inc"
  ```
- Contains routines by the author for I/O

`print_int`	prints **EAX**
`print_char`	prints ASCII value of **AL**
`print_string`	prints the string stored at the address of **EAX**; must be 0 terminated
`print_nl`	prints newline
`read_int`	reads an integer into **EAX**
`read_char`	reads a character into **EAX**

First program

```
;
; file: first.asm
; First assembly program. This program asks for two integers as
; input and prints out their sum.
;
; To create executable:
; Using djgpp:
; nasm -f coff first.asm
; gcc -o first first.o driver.c asm_io.o
;
; Using Linux and gcc:
; nasm -f elf first.asm
; gcc -o first first.o driver.c asm_io.o
;
; Using Borland C/C++
; nasm -f obj first.asm
; bcc32 first.obj driver.c asm_io.obj
;
; Using MS C/C++
; nasm -f win32 first.asm
; cl first.obj driver.c asm_io.obj
```

```
;
; Using Open Watcom
; nasm -f obj first.asm
; wcl386 first.obj driver.c asm_io.obj
%include "asm_io.inc"
;
; initialized data is put in the .data segment
;
segment .data
;
; These labels refer to strings used for output
;
prompt1 db    "Enter␣a␣number:␣", 0       ; don't forget nul terminato
prompt2 db    "Enter␣another␣number:␣", 0
outmsg1 db    "You␣entered␣", 0
outmsg2 db    "␣and␣", 0
outmsg3 db    ",␣the␣sum␣of␣these␣is␣", 0
;
; uninitialized data is put in the .bss segment
;
segment .bss
;
; These labels refer to double words used to store the inputs
```

```
;
input1  resd 1
input2  resd 1
;
; code is put in the .text segment
;
segment .text
        global  asm_main
asm_main:
        enter   0,0                 ; setup routine
        pusha
        mov     eax, prompt1        ; print out prompt
        call    print_string
        call    read_int            ; read integer
        mov     [input1], eax       ; store into input1
        mov     eax, prompt2        ; print out prompt
        call    print_string
        call    read_int            ; read integer
        mov     [input2], eax       ; store into input2
        mov     eax, [input1]       ; eax = dword at input1
        add     eax, [input2]       ; eax += dword at input2
        mov     ebx, eax            ; ebx = eax
        dump_regs 1                 ; dump out register values
        dump_mem 2, outmsg1, 1      ; dump out memory
;
; next print out result message as series of steps
;
        mov     eax, outmsg1
        call    print_string        ; print out first message
        mov     eax, [input1]
        call    print_int           ; print out input1
        mov     eax, outmsg2
        call    print_string        ; print out second message
        mov     eax, [input2]
        call    print_int           ; print out input2
        mov     eax, outmsg3
        call    print_string        ; print out third message
        mov     eax, ebx
        call    print_int           ; print out sum (ebx)
        call    print_nl            ; print new-line
        popa
        mov     eax, 0              ; return back to C
        leave
        ret
```

C driver

```
#include "cdecl.h"
int PRE_CDECL asm_main( void ) POST_CDECL;
int main()
{
  int ret_status;
  ret_status = asm_main();
  return ret_status;
}
```

- All segments and registers are initialized by the C system
- I/O is done through the C standard library
- Initialized data in **.data**
- Uninitialized data in `.bss` (block started symbol)
- Code in `.text`
- Stack segment later

Compiling

- `nasm -f elf first.asm`
  produces `first.o`
- ELF: executable and linkable format
- `gcc -c driver.c`
  - produces `driver.o`
  - option `-c` means compile only
- We need to compile `asm_io.asm`:
  `nasm -f elf -d ELF_TYPE asm_io.asm`
  - produces `asm_io.o`
- On 64-bit machines, add the option `-m32` to generate 32-bit code, e.g. `gcc -m32 -c driver.c`

Linking

- Linker: combines machine code & data in object files and libraries together to create an executable
- `gcc -o first driver.o first.o asm_io.o`
- On 64-bit machines,
  `gcc -m32 -o first driver.o first.o asm_io.o`
- `-o outputfile` specifies the output file
- `gcc driver.o first.o asm_io.o`
  produces `a.out` by default