

# Basic Program Structure (Asm + C)

Note Title

9/26/2007

Standard form ASM

label: opcode operands comment

labels name current position in code  
current pos. is addr at which next  
inst. goes

```
foo: add $1, %eax  
sub $1, %ebx
```

```
one foo ; br. to foo if zero not set
```

Directives are aimed at the assembler

```
.equiv MAX_CNT, 12  
movl $MAX_CNT, %ebx  
movl $12, %ebx  
pseudo-op  
MAX_CNT .equ 12
```

.text

tells assembler we are in text (code) section

other common section names include

.rodata	read only data
.data	initialized data
.bss	uninitialized data

comes from IBM 704 assembly lang. late 1950s  
"block storage start"

distinct sections useful in embedded  
systems

alternative syntax

\*section name

labels important in data section

```
.section .data  
i: .int 0  
str: .string "this is an example"  
var: .int 3
```

. is label that means "right here"  
 exp such as .+4 means 4 address beyond  
 current position

```
add    ---      xx xx xx xx
.org   .+8      ( 8 bytes empty )
sub    ---      xx xx xx xx
```

```
/*
 * initial set of comments describing
 * func. of prog. or file
 */
```

```
.text      // code section //
.global main

main:
    pushl   %ebp
    movl   %esp, %ebp
    pushl   %ebx
    pushl   %esi
    pushl   %edi

// assembly code to define function //
```

```
popl   %edi
popl   %esi
popl   %ebx
movl   %ebp, %esp
popl   %ebp
ret
```

```
.data
...
.bss
...
```

### C program structure

classically sep into two sets of files

```
___ .c    program files
___ .h    header files
```

```
/*
 * bar.h
 * global comments
 */
#define MAX_CNT 12 //preprocessor dir
```

```
/*  
 * bar.c  
 * header comment  
 */  
#include <stdio.h> // <> system  
#include "bar.h" // " " local  
  
.data int var = 1; // global vars  
.bss int x;  
main (...) {  
    int var = 2; // local vars  
  
    /* code */  
    x = var;  
}
```