Chapter 10
Mixing C and Assembly

Dr. Yifeng Zhu
Electrical and Computer Engineering
University of Maine

Spring 2015

Basic Data Types in C

Data Type	Data Size (bits)	Alignment	Data Range
bool	8	byte	0 or 1. Bits 1-7 are ignored
char	8	byte	-128 - 127(signed) or 0 - 255(unsigned)
int	32	word	-2,147,483,648 - 2,147,483,647(signed) or 0 - 4,294,967,296(unsigned)
short int	16	halfword	-32,768 - 32,767(signed) or 0 - 65,536(unsigned)
long int	32	word	same as int
long long	64	word	-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807(signed) or 0 - 18,446,744,073,709,551,616(unsigned)
float	32	word	+/- 1.4023×10^{-45} to $3.4028 \times 10^{+38}$, always signed
double	64	word	+/- 4.9406×10^{-324} to 1.7977×10^{308}, always signed
long double	96	word	very large range

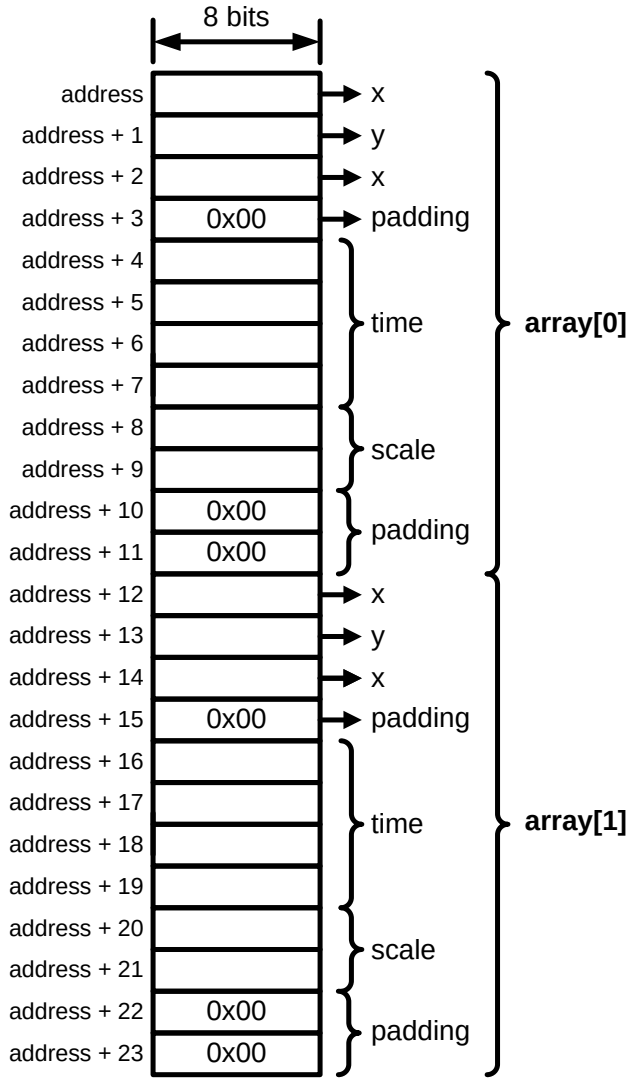
Load and Store Data

unsigned char	LDRB/STRB
unsigned short	LDRH/STRH
unsigned int	LDR/STR
char	LDRSB/STRSB
short	LDRSH/STRSH

Struct

```
struct Position {
  char x;
  char y;
  char x;
  int time;
  short scale;
} array[10];
```

```
struct Position {
  char x;
  char y;
  char x;
  char padding_1[1];
  int time;
  short scale;
  char padding_2[2];
} array[10];
```



Static Variables

C Code

```
int foo();

int main(void) {
   int y;
   y = foo();  // y = ?
   y = foo();  // y = ?
   y = foo();  // y = ?
   while(1);
}

int foo() {
   int x = 5;
   x = x + 1;
   return(x)
}
```

C Code

```
int foo();

int main(void) {
   int y;
   y = foo();  // y = ?
   y = foo();  // y = ?
   y = foo();  // y = ?
   while(1);
}

int foo() {
   static int x = 5;
   x = x + 1;
   return(x)
}
```

Static Variables

C Code	Assembly Code

`int foo();`
 	
`int main(void) {`
`  int y;`
`  y = foo();  // y = 6`
`  y = foo();  // y = 6`
`  y = foo();  // y = 6`
`  while(1);`
`}`
 	
`int foo() {`
`  int x = 5;  // local variable`
`  x = x + 1;`
`  return(x)`
`}`
	`  AREA static_demo, CODE`
	`  EXPORT __main`
	`  ALIGN`
	`  ENTRY`
	`__main  PROC`
	`  BL  foo  ; r0 = 6`
	`  BL  foo  ; r0 = 6`
	`  BL  foo  ; r0 = 6`
	`stop  B  stop`
	`  ENDP`
	
	`foo  PROC`
	`  MOV  r0,#5`
	`  ADD  r0,r0,#1`
	`  BX  LR`
	`  ENDP`
	`  END`

Static Variables

C Code	Assembly Code

`int foo();`
 	
`int main(void) {`
`  int y;`
`  y = foo();  // y = 6`
`  y = foo();  // y = 7`
`  y = foo();  // y = 8`
`  while(1);`
`}`
 	
`int foo() {`
`  // x is initialized only once`
`  static int x = 5; // a static variable`
`  x = x + 1;`
`  return(x)`
`}`
	`  AREA static_demo, CODE`
	`  EXPORT __main`
	`  ALIGN`
	`  ENTRY`
	`__main  PROC`
	`  BL  foo  ; r0 = 5`
	`  BL  foo  ; r0 = 6`
	`  BL  foo  ; r0 = 7`
	`stop  B  stop`
	`  ENDP`
	
	`foo  PROC`
	`  LDR  r1,=x`
	`  LDR  r0,[r1]`
	`  ADD  r0,r0,#1`
	`  STR  r0,[r1]`
	`  BX  LR`
	`  ENDP`
	
	`  AREA myData, DATA`
	`  ALIGN`
	`x  DCD  5`
	`  END`

Example of C Calling Assembly

C Program (main.c)

```
char str[25] = "Hello!";

extern void strlen(char* s);

int main(void){
    int i;
    i = strlen(str);
    while(1);
}
```

Assembly Program (strlen.s)

```
        AREA stringLength, CODE
        EXPORT strlen       ; make strlen visible
        ALIGN
strlen  PROC
        PUSH {r4, lr}       ; preserve r4 and lr
        MOV  r4, #0         ; initialize length
loop    LDRB r1, [r0, r4]   ; r0 = string address
        CBZ  r1, exit       ; branch if zero
        ADD  r4, r4, #1     ; length++
        B    loop           ; do it again
exit    MOV  r0, r4         ; place result in r0
        POP  {r4, pc}       ; exit
        ENDP
```

Example of Accessing C Variables

C Program (main.c)

```
int counter;

extern int getValue();
extern void setValue(int c);

void increment();

int main(void) {
   int c = 0;
   setValue(1);
   increment();
   c = getValue();
   while(1);
}

void increment(){
   counter += 2;
}
```

Assembly Program (count.s)

```
            AREA count, CODE
            IMPORT counter
            ALIGN
setValue    PROC
            EXPORT setValue
            LDR r1, =counter
            STR r0, [r1]
            BX  lr
            ENDP
getValue    PROC
            EXPORT getValue
            LDR r1, =counter
            LDR r0, [r1]
            BX  lr
            ENDP
increment   PROC
            EXPORT increment [WEAK]
            LDR r1, =counter
            LDR r0, [r1]
            ADD r0, r0, #1
            STR r0, [r1]
            BX  lr
            ENDP
            END
```

Example of Assembly Calling C

Assembly Program (main.s)	C Program (strlen.c)

AREA my_strlen, CODE
EXPORT __main
IMPORT strlen
ALIGN
ENTRY
 	
__main PROC
LDR r0, =str
BL strlen
stop B stop
ENDP
 	
AREA myData, DATA
ALIGN
Str DCB "12345678",0
END
	int strlen(char *s){
	int i = 0;
	
	while(s[i] != '\0')
	i++;
	
	return i;
	}

Example of Accessing Data Defined in Assembly

Assembly Program	C Program

AREA main, CODE
EXPORT __main
IMPORT getValue
IMPORT increment
IMPORT setValue
ALIGN
ENTRY
 	
__main MOVS r2,#0
MOVS r0,#1
BL setValue
BL increment
BL getValue
MOV r2,r0
stop B stop
 	
AREA myData, DATA
EXPORT counter
counter DCD 0
END
	extern int counter;
	
	int getValue() {
	return counter;
	}
	
	void increment() {
	counter++;
	}
	
	void setValue(int c) {
	counter = c;
	}