Example 9.3. Write software to output an ASCII string an output device.

Solution: Because the length of the string may be too long to place all the ASCII characters into the registers at the same time, call by reference parameter passing will be used. With call by reference, a pointer to the string will be passed. The function **`UART_OutString`**, shown in Program 9.5, will output the string data to the display. A version of the function **`UART_OutChar`** will be developed in Chapter 11 which sends data out the UART. When using a development kit, this UART data is observable on the PC running a terminal program like PuTTY or TExaSdisplay. For now all we need to know is that it outputs a single ASCII character. In the assembly version R4 is used because we know by convention the function **`UART_OutChar`** will preserve R4 to R11. However, by convention this function will preserve R4 by saving and restoring it. R4 is a pointer to the string; one is added to the pointer each time through the loop because each element in the string is one byte. Since this function calls a subfunction it must save LR. The POP PC operation will perform the function return.

```
;Input: R0 points to string
UART_OutString
     PUSH {R4, LR}
     MOV  R4, R0
loop LDRB R0, [R4]
     ADD  R4, #1  ;next
     CMP  R0, #0  ;done?
     BEQ  done    ;null?
     BL   UART_OutChar ;print character
     B    loop
done POP  {R4, PC}
```

```
// displays a string
void UART_OutString(char buf[]){
unsigned long i=0;
  while(buf[i]){
    UART_OutChar(buf[i]); // output
    i++;          // next
  }
}
```

Program 9.5. A variable length string contains ASCII data.

Observation: Most C compilers have standard libraries. If you include “string.h” you will have access to many convenient string operations.

When dealing with strings we must remember that they are arrays of characters with null termination. In C, we can pass a string as a parameter, but doing so creates a constant string and implements call by reference. Assuming **`Hello`** is as

defined above, these three invocations are identical:

```
OutString(Hello);
OutString(&Hello[0]);
OutString("Hello world\n\r");
```

Previously we dealt with constant strings. With string variables, we do not know the length at compile time, so we must allocate space for the largest possible size the string could be. E.g., if we know the string size could vary from 0 to 19 characters, we would allocate 20 bytes.

```
char String1[20];
char String2[20];
```

In C, we cannot assign one string to another. I.e., these are illegal

```
String1 = "Hello";   //********illegal************
String2 = String1;   //********illegal************
```

We can make this operation occur by calling a function called **`strcpy`**, which copies one string to another. This function takes two pointers. We must however make sure the destination string has enough space to hold the string being copied.

```
strcpy(String1,"Hello");  // copies "Hello" into String1
strcpy(String2,String1);  // copies String1 into String2
```

Help

Program 9.6 shows two implementations of this string copy function. R0 and R1 are pointers, and R2 contains the data as it is being copied. In this case, **`dest++;`** is implemented as an “add 1” because the data is one byte each. If the data were 16-bit halfwords, the increment pointer would be “add 2” . If the data were 32-bit words the increment pointer would be “add 4”.

```
; Input: R0=&dest    R1=&source
strcpy LDRB R2,[R1]  ;source data
       STRB R2,[R0]  ;copy
       CMP  R2,#0    ;termination?
       BEQ  done
       ADD  R1,#1    ;next
       ADD  R0,#1
       B    strcpy
done   BX   LR

;faster version
strcpy LDRB R2,[R1],#1 ;source data
       STRB R2,[R0],#1 ;copy
       CMP  R2,#0      ;termination?
       BEQ  done
       B    strcpy
done   BX   LR
```

```
// copy string from source to dest
void strcpy(char dest[], char source[]){
unsigned long i=0;
  while(source[i]){
    dest[i] = source[i];  // copy
    i++;           // next
  }
  dest[i] = 0;   // termination
}

// another version, using pointers
void strcpy(char *dest, char *source){
char data;
  do{
   data = *dest++ = *source++;
  } while(data);
}
```

Program 9.6. Simple string copy functions.

About (https://www.edx.org/about-us) Jobs (https://www.edx.org/jobs)
Press (https://www.edx.org/press) FAQ (https://www.edx.org/student-faq)
Contact (https://www.edx.org/contact)

EdX is a non-profit created by founding partners Harvard and MIT whose mission is to bring the best of higher education to students of all ages anywhere in the world, wherever there is Internet access. EdX's free online MOOCs are interactive and subjects include computer science, public health, and artificial intelligence.

(http://www.meetup.com/edX-Global-Community/)

(http://www.facebook.com/EdxOnline)

(https://twitter.com/edXOnline)

(https://plus.google.com/108235383044095082735/posts)

(http://youtube.com/user/edxonline)

© 2014 edX, some rights reserved.

Terms of Service and Honor Code - Privacy Policy (https://www.edx.org/edx-privacy-policy)

Help