Mixing C/C++ and Assembly Using SingleStep

The SingleStep software is able to compile C and C++ code for use with PowerPC processors. However, it is sometimes beneficial to use both C and ASM in the same program. This document will explain how to write, compile, and link programs that use a combination of C and ASM.

Since the C/C++ compiler and the assembler are two separate programs, you will have to create separate files for your C code and your assembly. You can then write functions in C that are called by assembly, and functions in assembly that are called by C.

Writing and compiling the assembly modules

No changes need to be made to the body of an assembly module in order to call C functions. Call your C function like you would any assembly function using the branch and link instruction, **bl**. For example, if your C/C++ function is declared as `int myFunc(...)`, use:

```
bl myFunc
```

To call ASM functions from C, make the label of the ASM function global and word-align it:

```
.align 2        #word-align next value
.globl myFunc   #make myFunc global
myFunc:         #function starts here
```

To assemble your ASM files, you may want to create a batch file **makeasm.bat** with the following command:

```
asppc -c -g -l %1.s > %1.lst
```

Assemble the file myFile.s by using:

```
makeasm myFile
```

Writing and compiling the C/C++ modules

Your C/C++ modules will essentially be a set of functions that are called by the main assembly body. For C programs, functions can be declared as normal. For C++ functions, you must prefix function declarations with `extern "C"` or the compiler will mangle the name and you will be unable to access the function from your assembly code.

C function:	`int myFunc(...)`
C++ function:	`extern "C" int myFunc(...)`

To call ASM functions from your C++ code, use the ASM function's label as the function name and call it like any other C/C++ function:

```
/* function myFunc is an assembly function
that returns an integer value */

int b;

b = myFunc(...);
```

To compile your C/C++ programs, you may want to create a batch file **makec.bat** with the following command:

```
hcppc -c -g %1.c
```

Assemble the file myFile.c by using:

```
makec myFile
```

The compiler is available in the lab under c:\SDS75\hcppc\bin. The C compiler will *not* work on CAEN machines. The compiler is also available in the demo version of the SingleStep software on the course web page. Once installed, it is under c:SDS74\hcppc\bin.

ABI Conventions

C/C++ functions compiled using the hcppc compiler will follow the PowerPC ABI conventions. For details on the PowerPC ABI, refer to the **EABI** document in the **References / Handouts** section of the course website. Specifically, the following conventions must be observed:

- Parameters passed to C/C++ functions from assembly must be placed in r3-r10 in the order that the parameters appear in the function declaration. Similarly, parameters passed to assembly functions from C will appear in r3-r10 in the order that they were passed.
- Return values must be placed in r3-r4 when returning to C, and will appear in r3-r4 when returning to assembly.
- The stack pointer must be in r1.

Linking files

Object files (.o) will be generated when you compile your C and ASM code. To link the .o files into an executable .elf file, create and use the following batch file, **link.bat**:

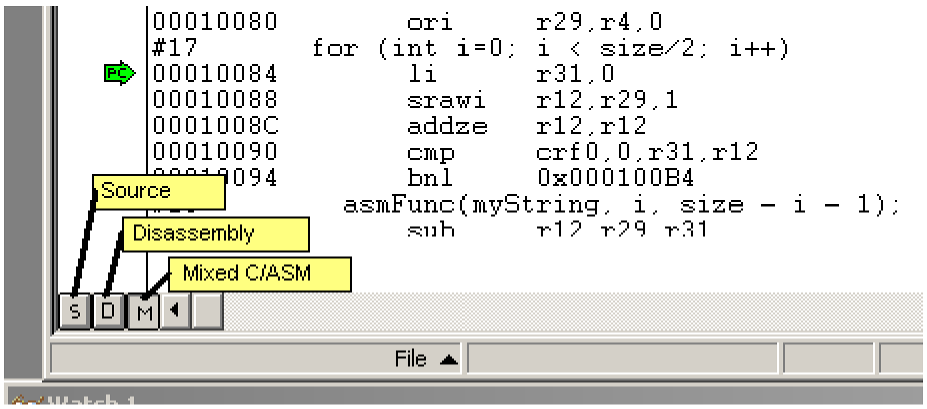
```
ldppcl -dn -A  %1-lnk.txt -o %1.elf %~2
```

Link the files by using:

```
        link myproj "asmfile1.o asmfile2.o cfile1.o cfile2.o"
```

where "myproj" is the name used in your link file, i.e. myproj-lnk.txt, and the list of object files you want to link is enclosed in quotes.

Debugging mixed C/ASM in SingleStep



```
00010080        ori     r29,r4,0
#17         for (int i=0; i < size/2; i++)
00010084        li      r31,0
00010088        srawi   r12,r29,1
0001008C        addze   r12,r12
00010090        cmp     crf0,0,r31,r12
[illegible]094        bnl     0x000100B4
[illegible]           asmFunc(myString, i, size - i - 1);
                [illegible]
```

SingleStep allows you to view your code in mixed C/ASM mode. Your C code will be displayed as assembler comments, and the assembly generated for each line will follow. To view your code in mixed mode, click on the 'M' button in the lower left corner of the debug window. You can set breakpoints and step through the compiler-generated assembly just like you would with your own assembly code.

Other things to note

- A sample program using mixed C/ASM has been provided on the website. Compile and link the modules, then simulate with SingleStep to see an example.

- Your main program must be in assembly. If you use C with a main() function, SingleStep will not be able to handle the final executable. Later updates may explain how to resolve this issue.