Subject: How to link assembler compiled file
Posted by frederik.dumarey on Sun, 18 May 2025 09:51:20 GMT
View Forum Message <> Reply to Message

Hello,

I added a assembler file named my_strlen.S to my package, the contents of this file are below:

```
 .global _my_strlen   # Export the symbol _my_strlen
 .text      # code section

_my_strlen:

 #function prologue
 pushq %rbp
 movq %rsp, %rbp
 movq %rdi, %rsi    # copy the string pointer to %rsi
 xorq %rax, %rax    # zero %rax to use as a counter

strlen_loop:
 cmpb $0, (%rsi)    # compare byte at %rsi with 0 (null terminator)
 je strlen_end    # if zero, end of string reached
 incq %rax     # increment counter
 incq %rsi     # move to next character
 jmp strlen_loop    # repeat loop

strlen_end:

 #function epilogue
 popq %rbp
 ret
```

Since this is AT&T assembler code, it compiled without any problem to a object file using right_click, build and compile option in TheIDE GUI. So far so good.

I then added a C++ file, which has the following content:

```
#include <iostream>

//declare the assembler function
extern "C" size_t my_strlen(const char* str);

int main(int argc, const char *argv[])
{
 const char* message = "Hello from Assembler";
 size_t length = my_strlen(message);
```

```
 std::cout << "Message: " << message << std::endl;
 std::cout << "Length: " << length << std::endl;
 return 0;
}
```

which can also be compiled using the same method as stated before.

Now my question: how do I link those two object files in the GUI? I suppose I have to go to Project menu item, then Custom build steps, but what do I enter in all these fields?

Thanks.

Subject: Re: How to link assembler compiled file
Posted by mirek on Thu, 22 May 2025 07:59:24 GMT
View Forum Message <> Reply to Message

frederik.dumarey wrote on Sun, 18 May 2025 11:51Hello,

I added a assembler file named my_strlen.S to my package, the contents of this file are below:

```
 .global _my_strlen   # Export the symbol _my_strlen
 .text      # code section

_my_strlen:

 #function prologue
 pushq %rbp
 movq %rsp, %rbp
 movq %rdi, %rsi    # copy the string pointer to %rsi
 xorq %rax, %rax    # zero %rax to use as a counter

strlen_loop:
 cmpb $0, (%rsi)    # compare byte at %rsi with 0 (null terminator)
 je strlen_end    # if zero, end of string reached
 incq %rax     # increment counter
 incq %rsi     # move to next character
 jmp strlen_loop    # repeat loop

strlen_end:

 #function epilogue
 popq %rbp
 ret
```

Since this is AT&T assembler code, it compiled without any problem to a object file using right_click, build and compile option in TheIDE GUI. So far so good.

I then added a C++ file, which has the following content:

```
#include <iostream>

//declare the assembler function
extern "C" size_t my_strlen(const char* str);

int main(int argc, const char *argv[])
{
 const char* message = "Hello from Assembler";
 size_t length = my_strlen(message);
 std::cout << "Message: " << message << std::endl;
 std::cout << "Length: " << length << std::endl;
 return 0;
}
```

which can also be compiled using the same method as stated before.

Now my question: how do I link those two object files in the GUI? I suppose I have to go to Project menu item, then Custom build steps, but what do I enter in all these fields?

Thanks.

Well, this is very exotic issue so it made me check the code and interestingly gcc builder (used with clang) simply treats .s files just like any other source (.c, .mm), including adding the .o to the linker. To be sure, you can check commandlines with Verbose.

Subject: Re: How to link assembler compiled file
Posted by frederik.dumarey on Sat, 24 May 2025 10:07:21 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Thanks for mentioning the verbose console check. The .S and .CPP files were correctly compiled in .O files, but the linking failed on the global export. Most stupid error: i forgot the underscore for my variable :blush: .

```
#include <iostream>

//declare the assembler function
extern "C" size_t _my_strlen(const char* str);
```

```
int main(int argc, const char *argv[])
{
 const char* message = "Hello from Assembler";
 size_t length = _my_strlen(message);
 std::cout << "Message: " << message << std::endl;
 std::cout << "Length: " << length << std::endl;
 return 0;
}
```

When you compile this all links up fine and runs perfectly!

So for those of you that want to include native assembler functions for SS2, AVX SIMD instructions, you can give it a try with this method :)
As Mirek mentioned, bit exotic, but I thought why not...

Have a nice day all of you,

Subject: Re: How to link assembler compiled file
Posted by mirek on Sat, 24 May 2025 11:55:10 GMT

View Forum Message <> Reply to Message

frederik.dumarey wrote on Sat, 24 May 2025 12:07Hello Mirek,

Thanks for mentioning the verbose console check. The .S and .CPP files were correctly compiled in .O files, but the linking failed on the global export. Most stupid error: i forgot the underscore for my variable :blush: .

```
#include <iostream>

//declare the assembler function
extern "C" size_t _my_strlen(const char* str);

int main(int argc, const char *argv[])
{
 const char* message = "Hello from Assembler";
 size_t length = _my_strlen(message);
 std::cout << "Message: " << message << std::endl;
 std::cout << "Length: " << length << std::endl;
 return 0;
}
```

When you compile this all links up fine and runs perfectly!

So for those of you that want to include native assembler functions for SS2, AVX SIMD

instructions, you can give it a try with this method :)
As Mirek mentioned, bit exotic, but I thought why not...

Just to be sure, are you aware there are intrinsics for these?

(And also that we have common NEON/SSE2 subset supported?)

Mirek

Subject: Re: How to link assembler compiled file
Posted by frederik.dumarey on Sat, 24 May 2025 18:12:32 GMT
View Forum Message <> Reply to Message

Quote:
Just to be sure, are you aware there are intrinsics for these?

Yes, I do, and for those interested, I have a small example of it here:

```
#include <iostream>
#include <immintrin.h>

int main(int argc, const char *argv[])
{
 alignas(32) float a[8] = {1.0f, 2.0f, 3.0f, 4.0f, 1.5f, 2.5f, 3.5f, 4.5f};
 alignas(32) float b[8] = {5.0f, 6.0f, 7.0f, 8.0f, 5.5f, 6.5f, 7.5f, 8.5f};

 //load data in AVX registers
 __m256 vec_a = _mm256_load_ps(a);
 __m256 vec_b = _mm256_load_ps(b);

 //multiply elements
 __m256 vec_mul = _mm256_mul_ps(vec_a, vec_b);

 //horizontal add to compute the sum of all elements
 __m256 temp = _mm256_hadd_ps(vec_mul, vec_mul);
 temp = _mm256_hadd_ps(temp, temp);

 //extract 128 lower bits and sum
 __m128 low = _mm256_castps256_ps128 (temp);
 __m128 high = _mm256_extractf128_ps (temp,1);
 __m128 sum = _mm_add_ps (low, high);

 //extract the final result
```

```
 float result = _mm_cvtss_f32 (sum);

 std::cout << "Dot product: " << result << std::endl;

 return 0;
}
```